Programming for Android - more than basics

Tomáš Kypta

@TomasKypta

Agenda

- Material design
- UX design patterns
- Dialogs
- Notifications
- Broadcast receiver
- Service

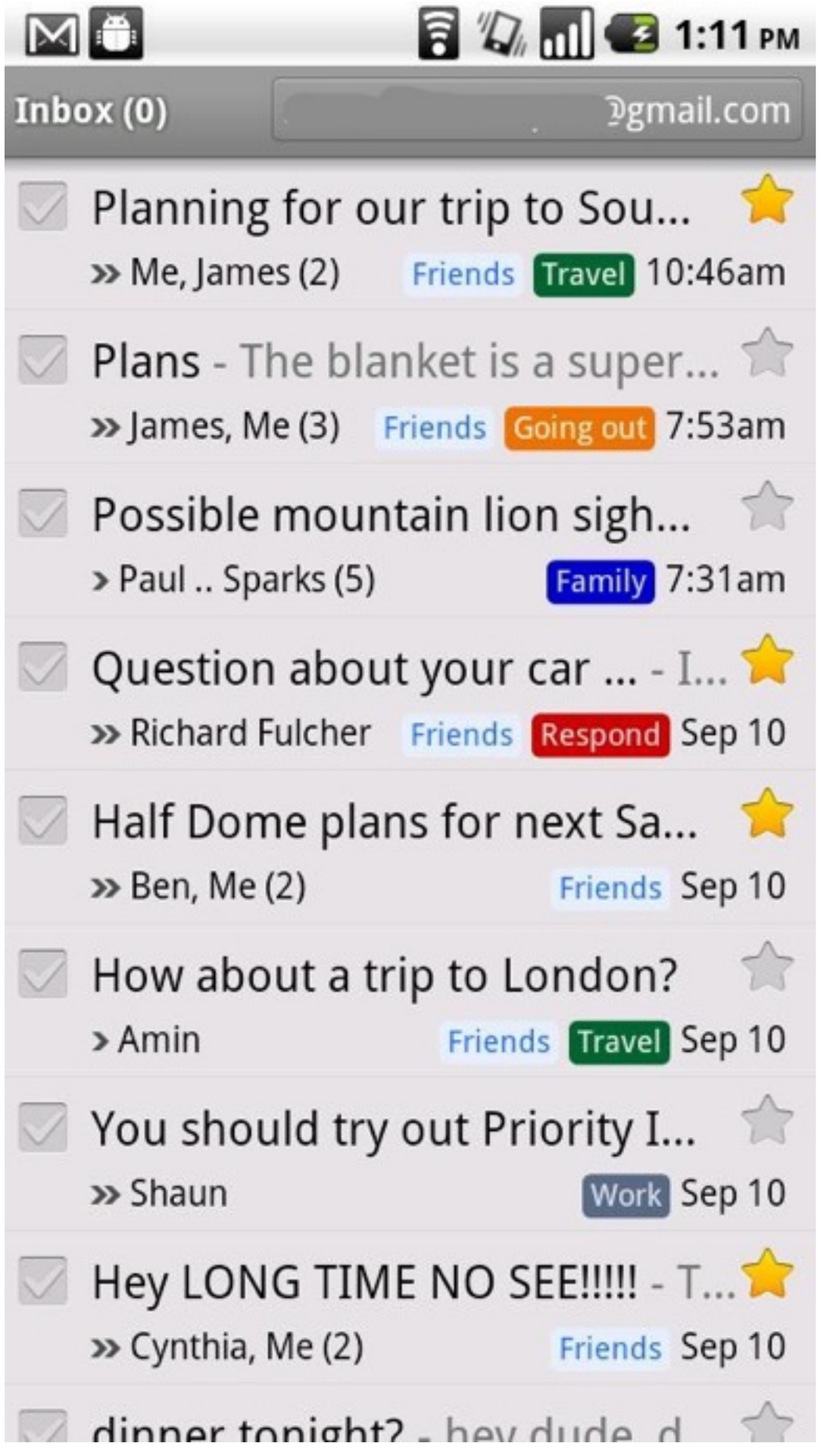
Agenda

- Fragment
- Process & memory
- Background processing
- Loader
- Software design patterns

Material Design

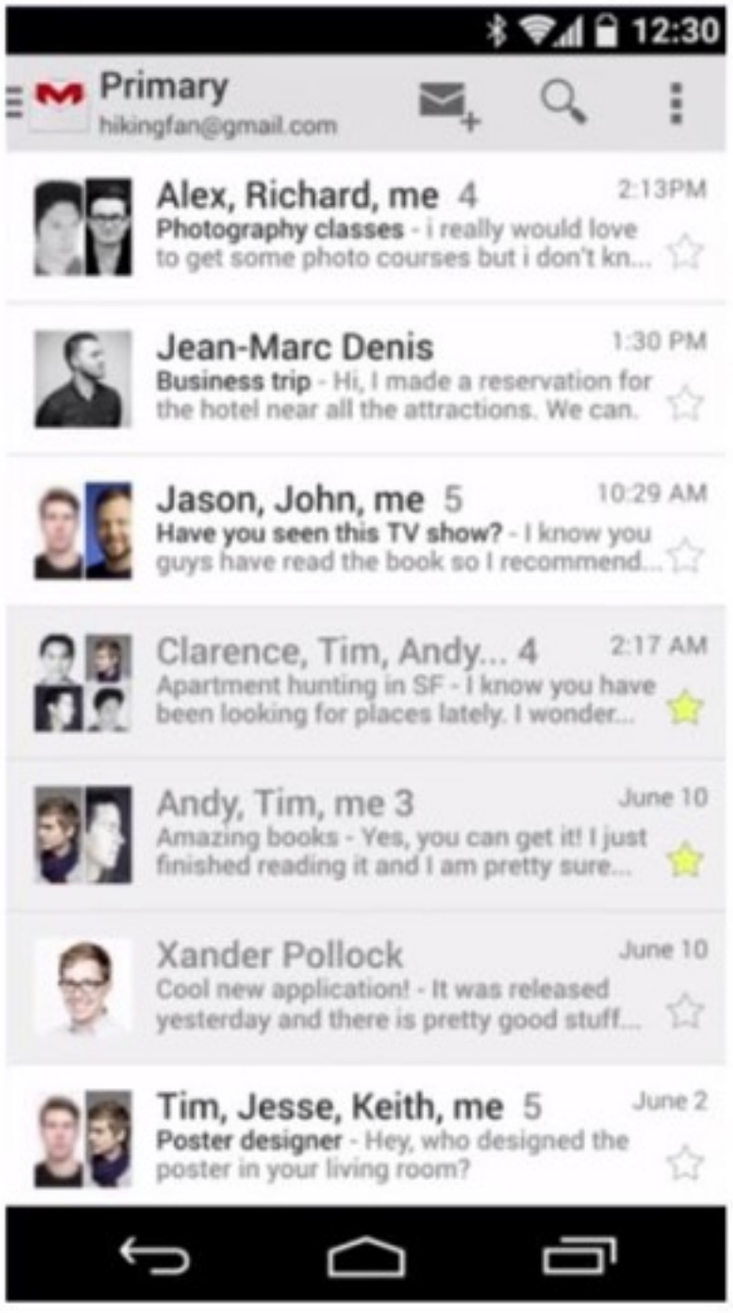
Android Design Evolution

long time ago …

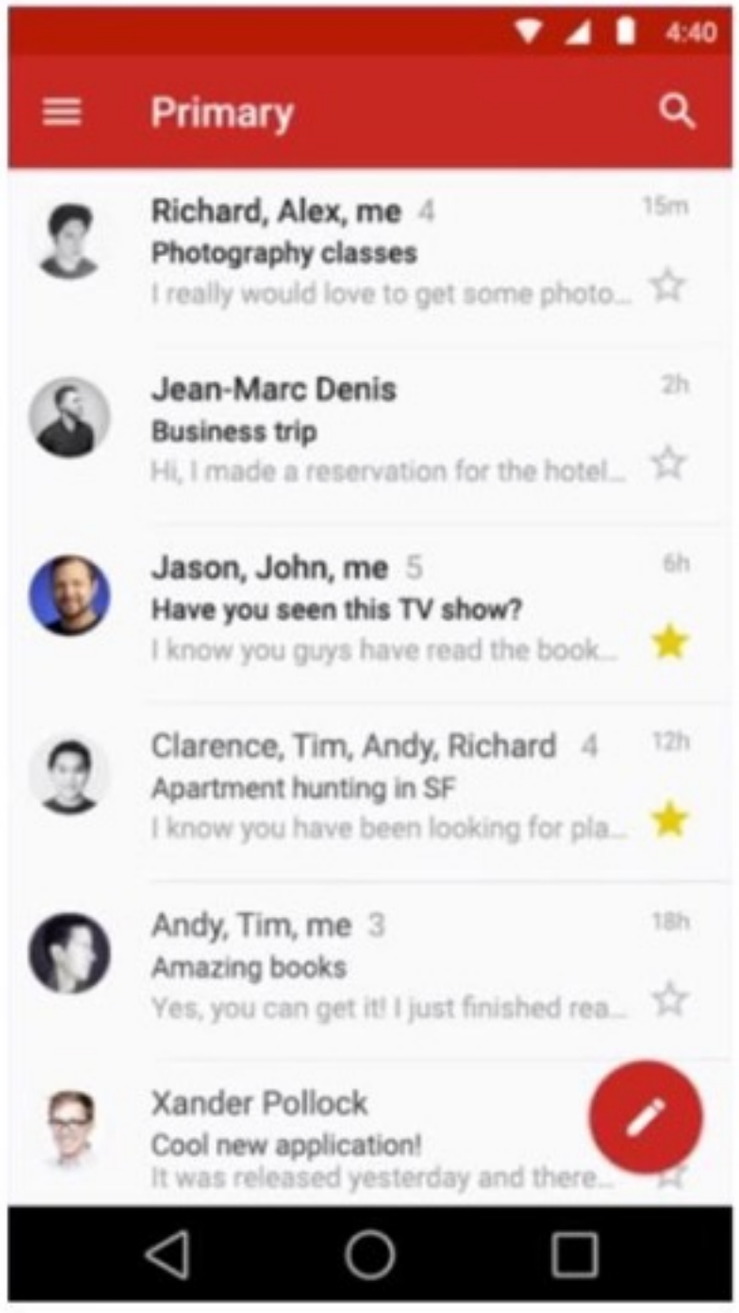


Android Design Evolution

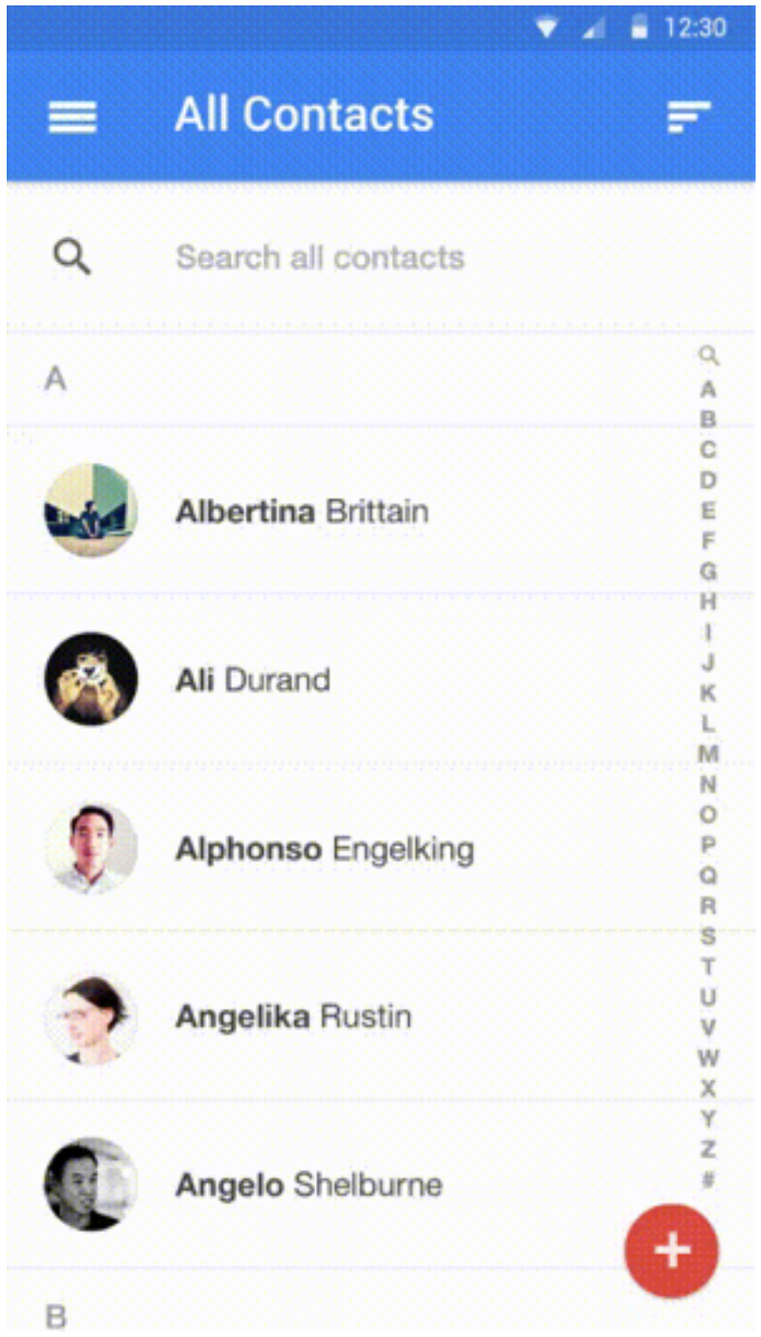
Holo Design

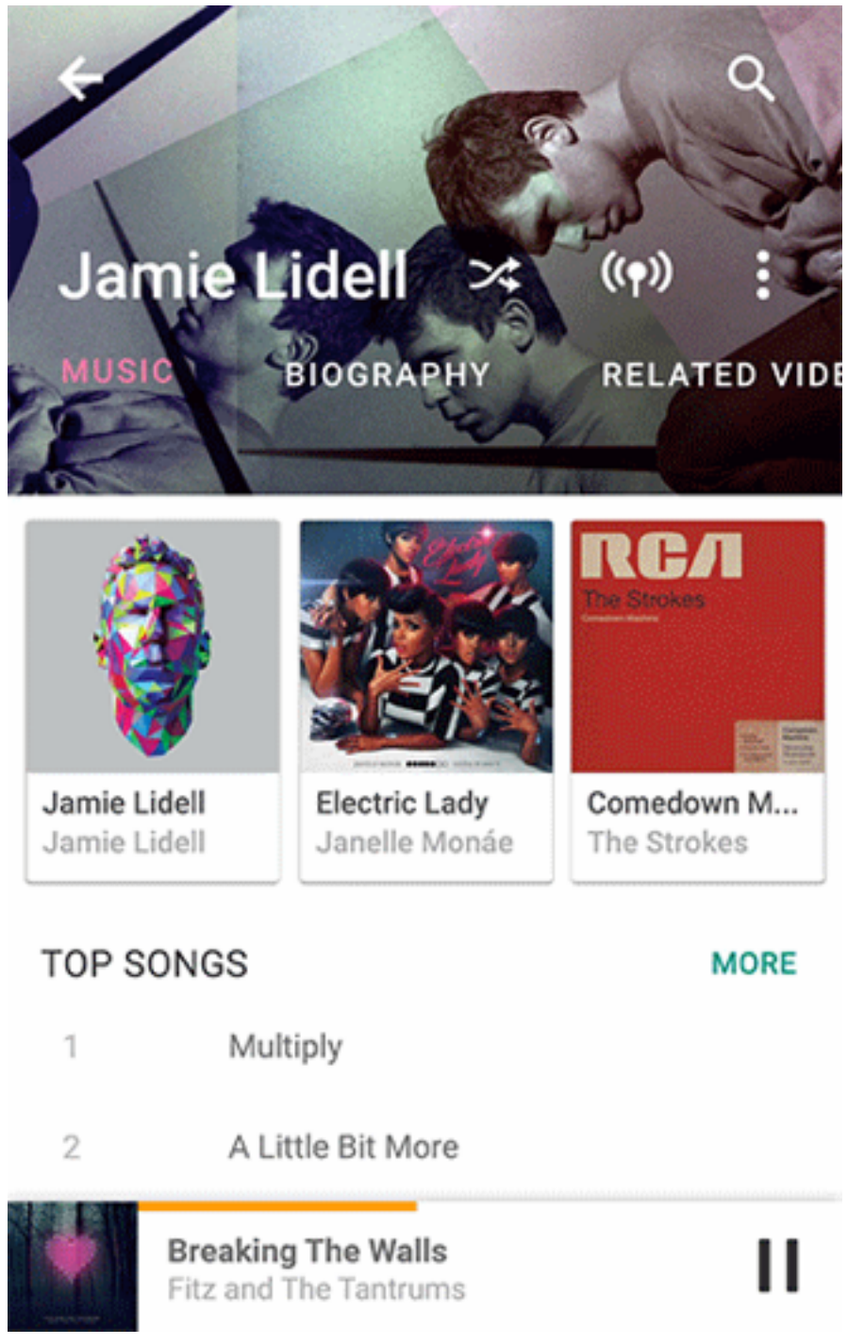


Material Design



Material Design





Material Design

- tangible surfaces and edges
  - shadows
  - elevation
- motion

Material Design

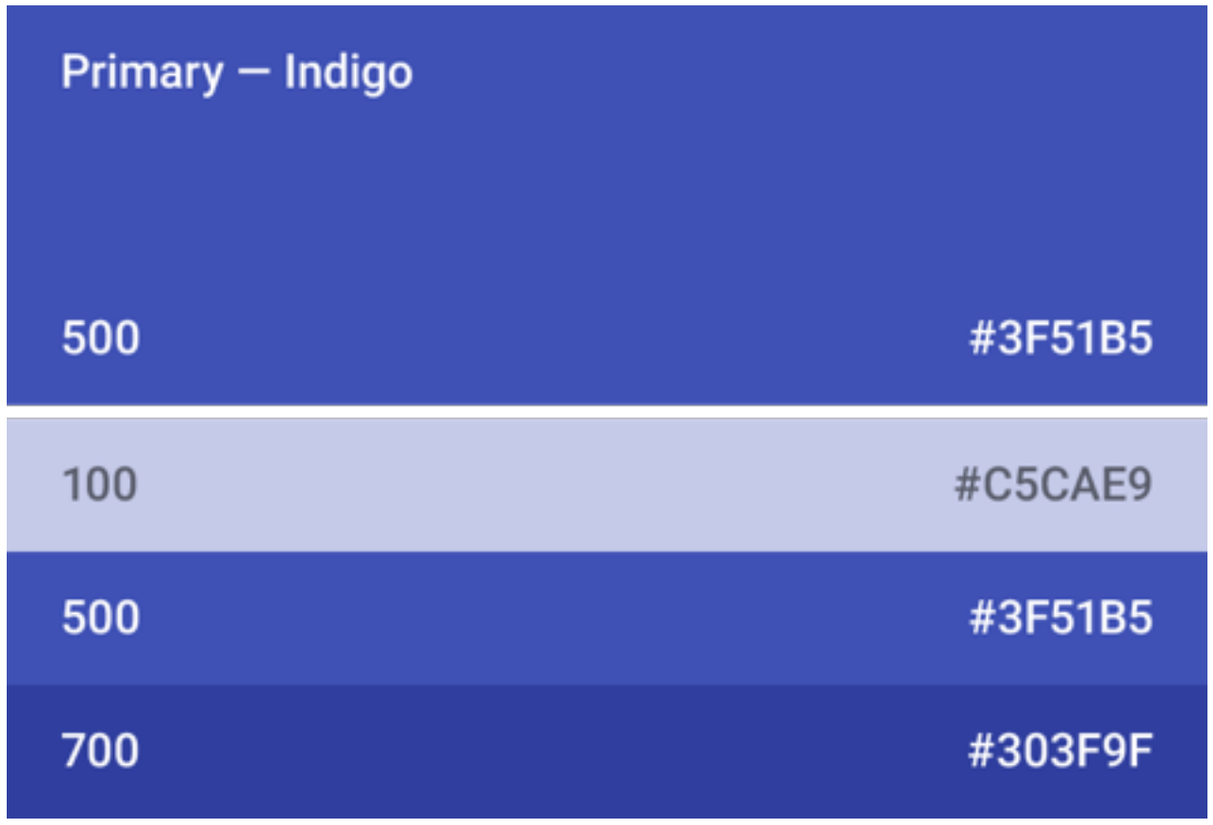
- themes since API level 21
  - `@android:style/Theme.Material` (dark version)
  - `@android:style/Theme.Material.Light` (light version)
  - `@android:style/Theme.Material.Light.DarkActionBar`
  - …

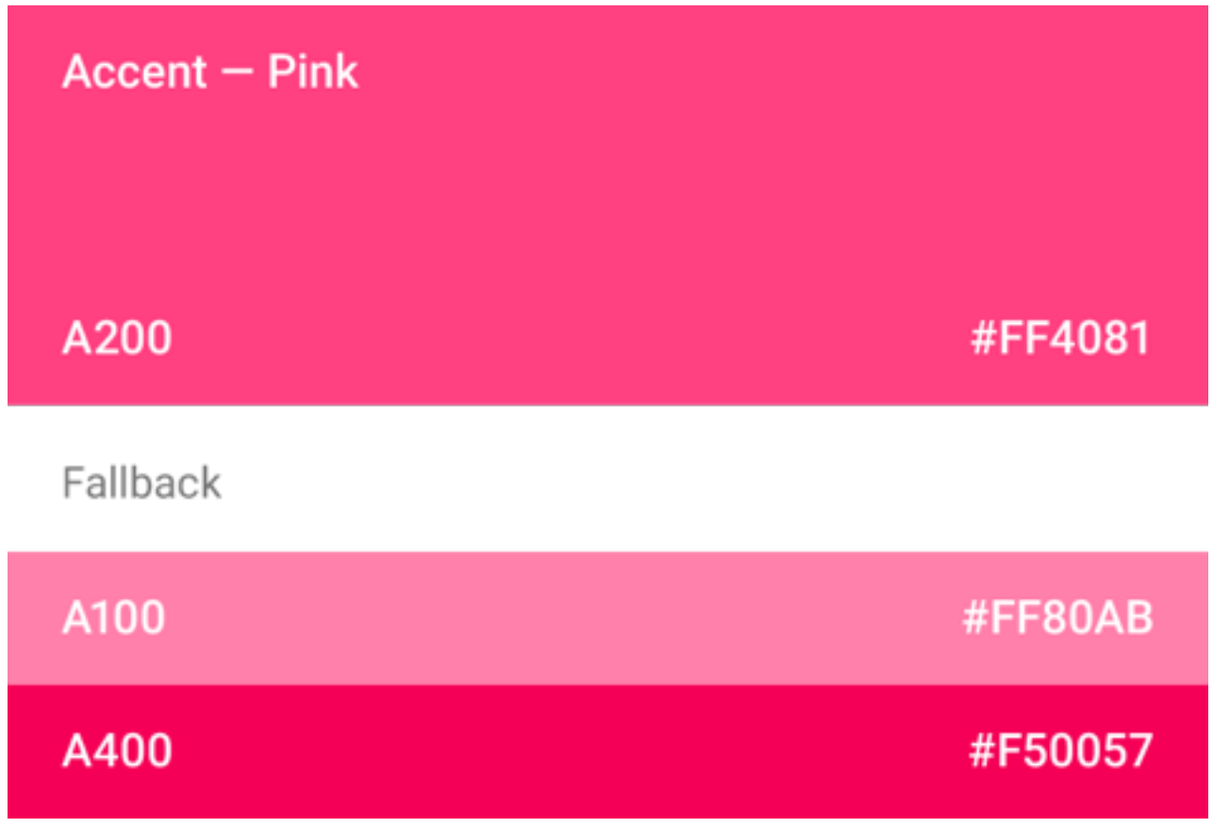
Material Design

- compatibility themes
  - `@style/Theme.AppCompat` (dark version)
  - `@style/Theme.AppCompat.Light` (light version)
  - `@style/Theme.AppCompat.Light.DarkActionBar`
  - …

The Color Palette

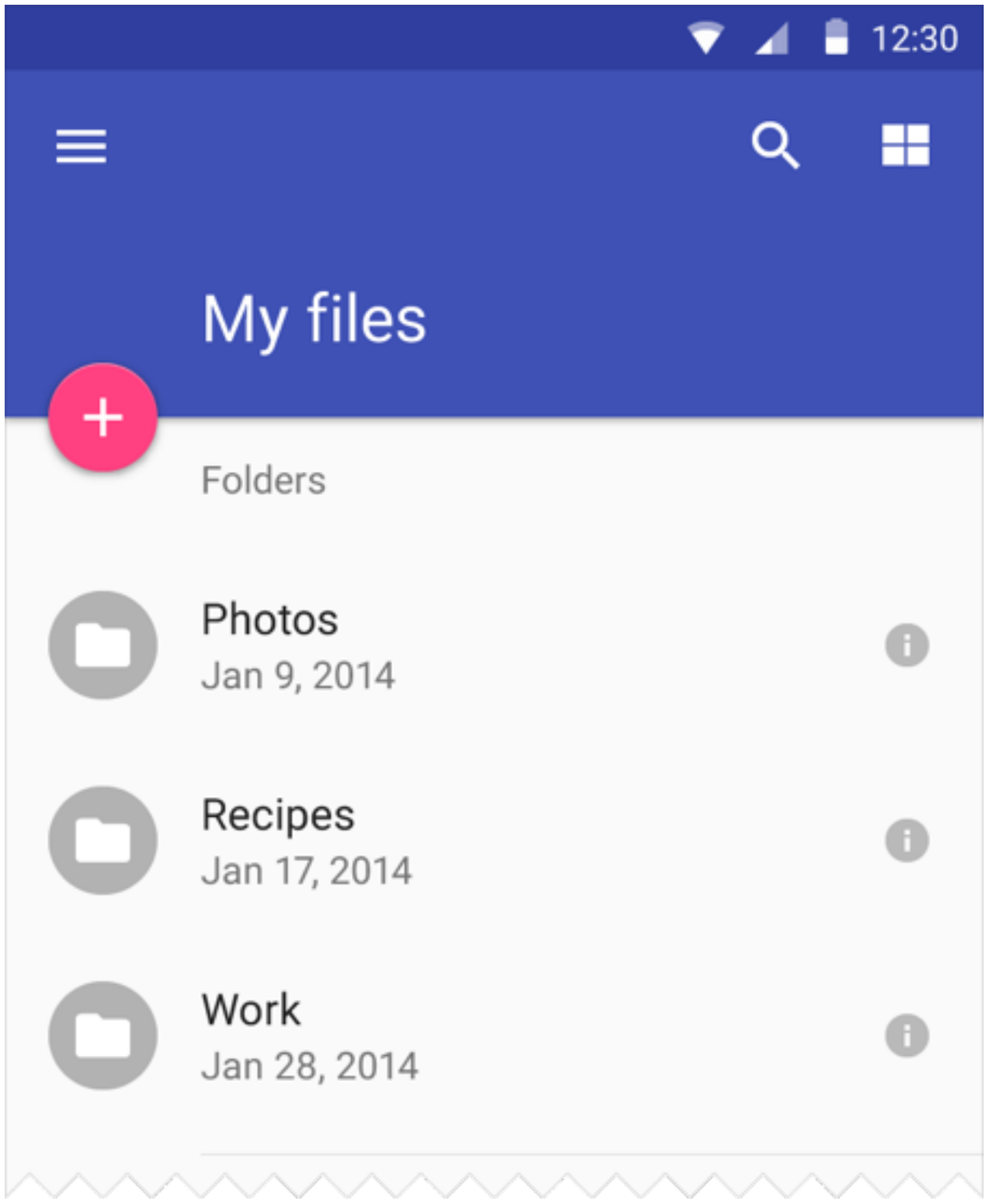
- three color hues from primary palette
- one accent color from the secondary palette





The Color Palette

- http://www.google.com/design/spec/style/color.html



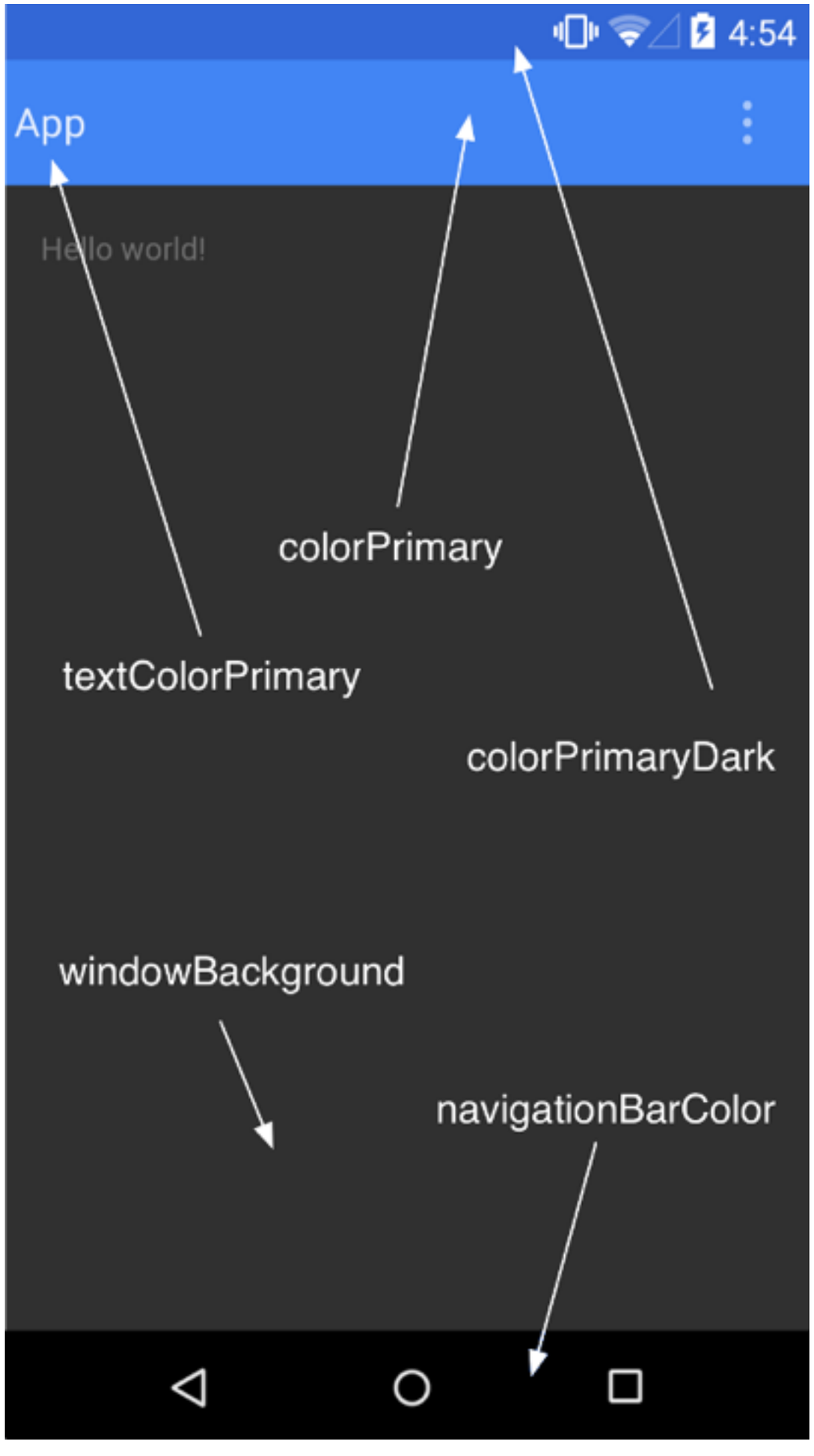
The Color Palette

```
<!-- inherit from the material theme -->
<style name="AppTheme" parent="“android:Theme.Material">

    <!-- Main theme colors -->
    <!-- your app branding color for the app bar -->
    <item name="“android:colorPrimary">@color/primary</item>

    <!-- darker variant for the status bar and contextual app bars -->
    <item name="“android:colorPrimaryDark">@color/primary_dark</item>

    <!-- theme UI controls like checkboxes and text fields -->
    <item name="android:colorAccent">@color/accent</item>
</style>
```

The Color Palette

```
<!-- AppCompat variant -->
<style name="AppTheme" parent="Theme.AppCompat">

    <!-- your app branding color for the app bar -->
    <item name="colorPrimary">@color/primary</item>

    <!-- darker variant for the status bar and contextual app bars -->
    <item name="colorPrimaryDark">@color/primary_dark</item>

    <!-- theme UI controls like checkboxes and text fields -->
    <item name="colorAccent">@color/accent</item>
</style>
```

The Color Palette



Toolbar

Toolbar

- generalization of action bar
- main Android navigation element

Toolbar

- `ActionBarActivity`

```
compile 'com.android.support:appcompat-v7:22.0.0'
```

- inherit styles from **`Theme.AppCompat`**
- for inflating views for action bar use **`getSupportActionBar().getThemedContext()`**

Toolbar

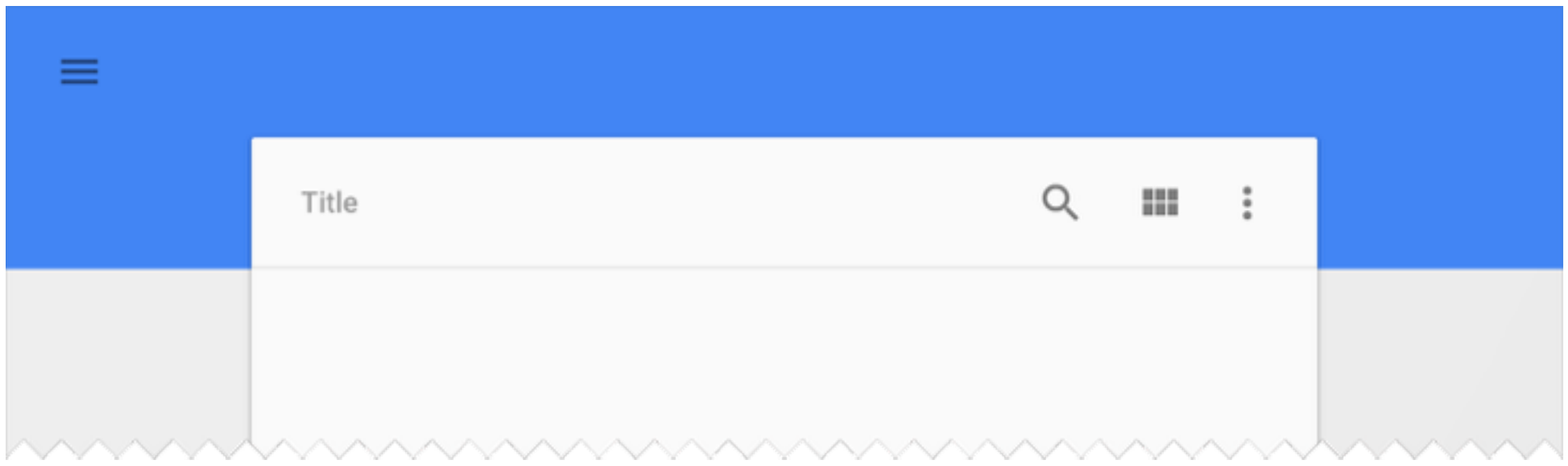
```
<android.support.v7.widget.Toolbar
android:id="@+id/my_toolbar"
android:layout_height="wrap_content"
android:layout_width="match_parent"
android:minHeight="?attr/actionBarSize"
android:background="?attr/colorPrimary" />
```

Toolbar

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    Toolbar toolbar = (Toolbar)
        findViewById(R.id.my_toolbar);
    setSupportActionBar(toolbar);
}
```

Toolbar

- standalone mode
- ~~`setSupportActionBar(toolbar);`~~



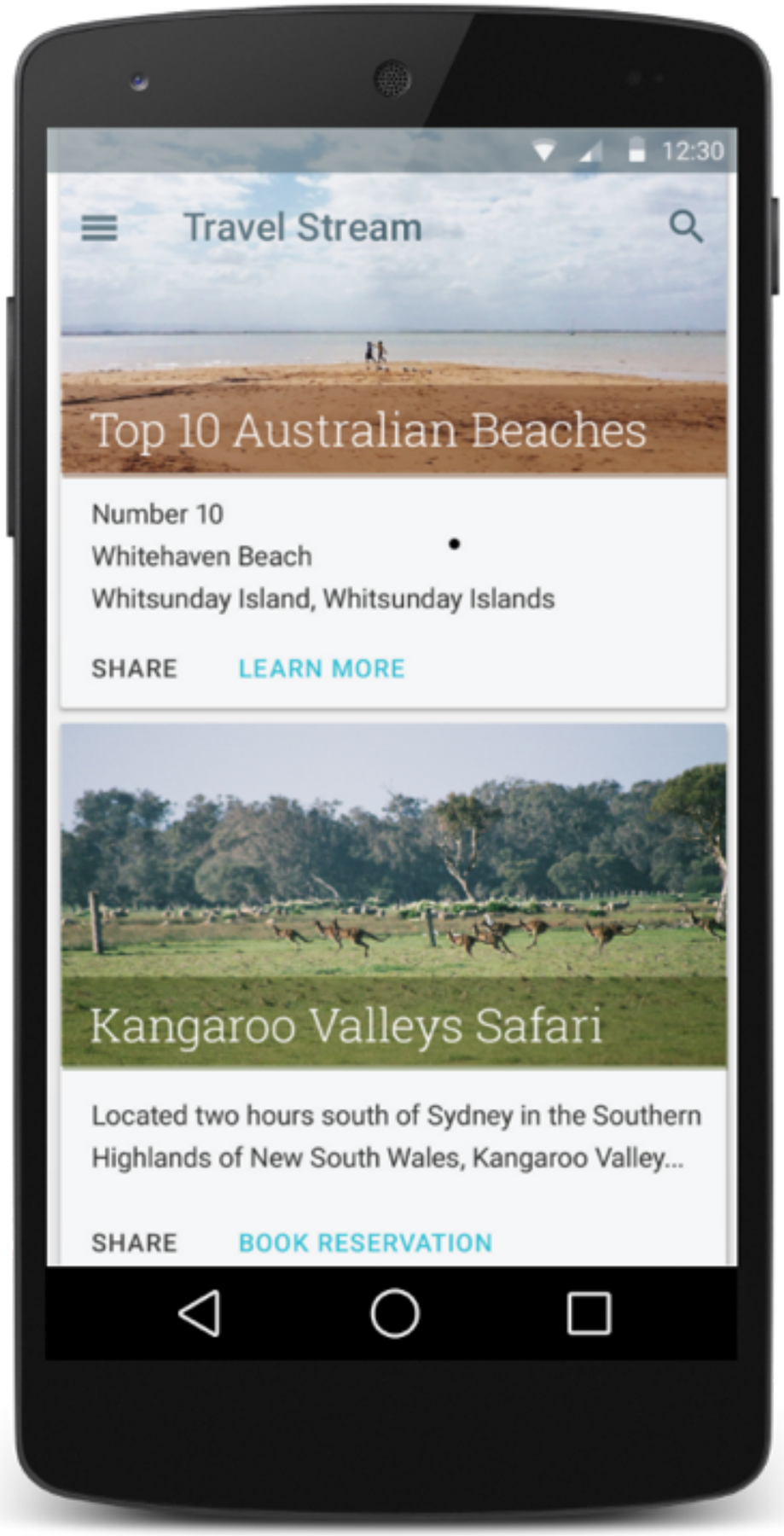
CardView

CardView

- showing information inside cards with consistent look

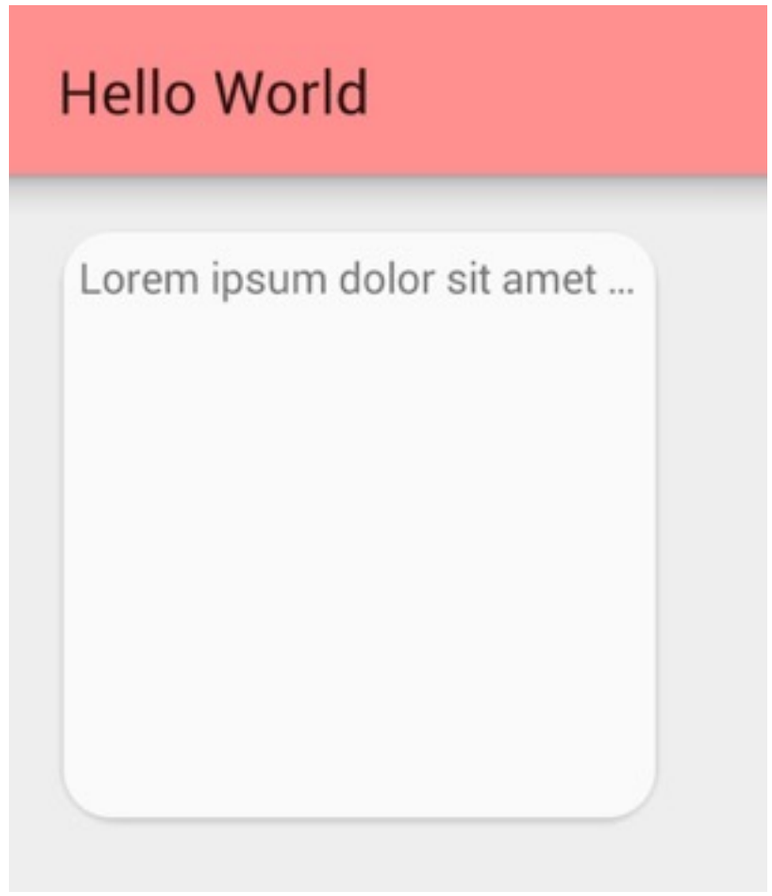
```
compile 'com.android.support:cardview-v7:22.0.0'
```

CardView


12:30
Travel Stream
Top 10 Australian Beaches
Number 10
Whitehaven Beach
Whitsunday Island, Whitsunday Islands
SHARE
LEARN MORE
Kangaroo Valleys Safari
Located two hours south of Sydney in the Southern Highlands of New South Wales, Kangaroo Valley...
SHARE
BOOK RESERVATION

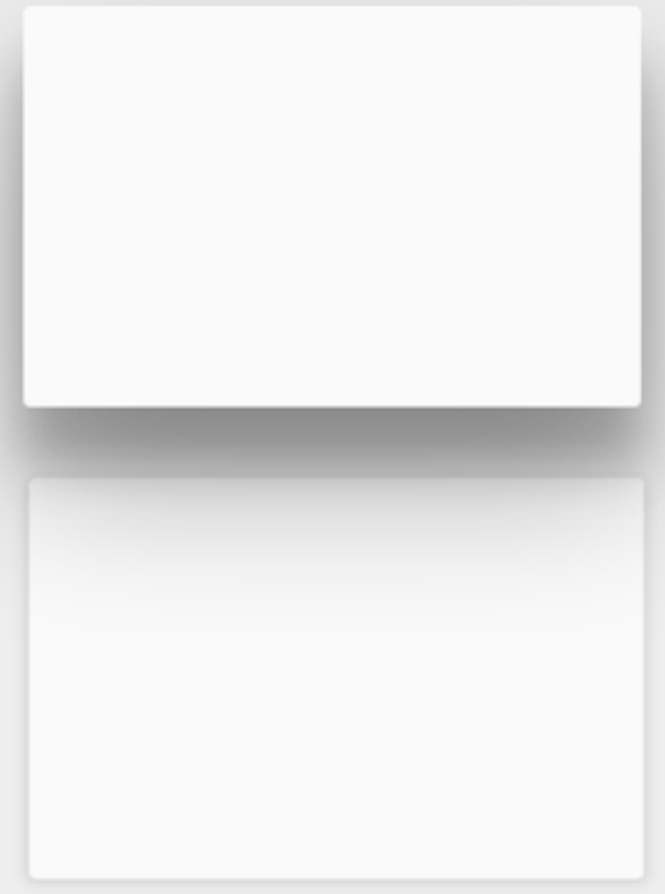
CardView

```
<android.support.v7.widget.CardView
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:id="@+id/card_view"
    android:layout_gravity="center"
    android:layout_width="200dp"
    android:layout_height="200dp"
    card_view:cardCornerRadius="16dp">

    …

</android.support.v7.widget.CardView>
```



Elevation

Elevation



Elevation

```
android:elevation=“4dp”

view.setElevation(elevation);
```

Transition

Transition Animation

- since API level 21

```
<item name=“android:windowEnterTransition">
        @android:transition/explode</item>
<item name=“android:windowExitTransition">
        @android:transition/explode</item>
```

Transition

```
public class MyActivity1 extends ActionBarActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // enable transitions before content is added
        getWindow().requestFeature(
            Window.FEATURE_CONTENT_TRANSITIONS);

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_my_1);
        …
    }
}
```

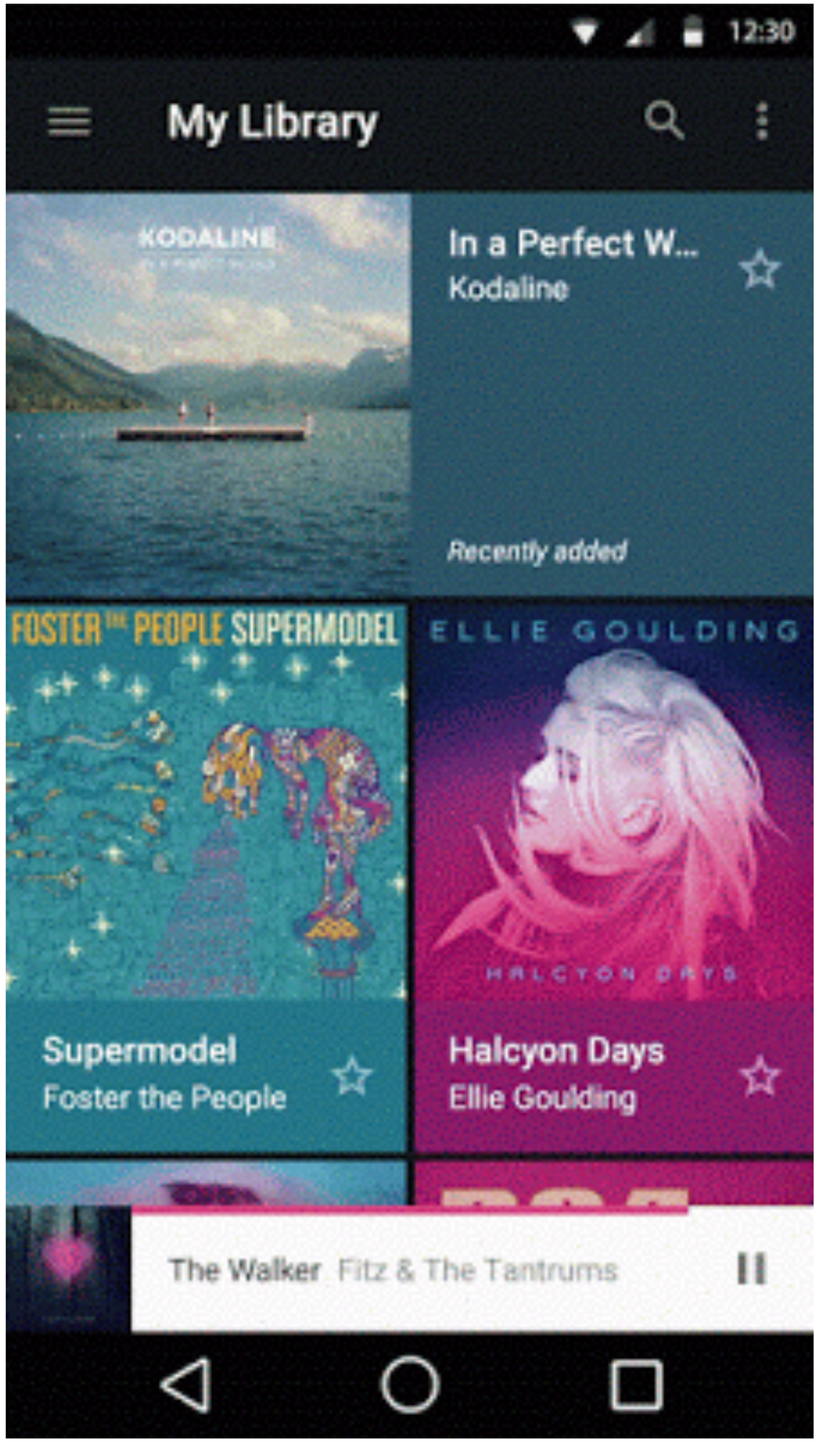
Transition

```
btn.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
            getWindow().setExitTransition(new Explode());
            Intent intent = new Intent(MyActivity1.this,
                MyActivity2.class);
            startActivity(intent,
                ActivityOptions
                    .makeSceneTransitionAnimation(MyActivity1.this)
                    .toBundle());
      }
});
```

Shared Element Transition

- enable window content transitions
- transition for shared element
  - define shared element with `android:transitionName`
    - in both layouts
  - `ActivityOptions.makeSceneTransitionAnimation()`

Shared Element Transition



Ripple

Ripple

- RippleDrawable

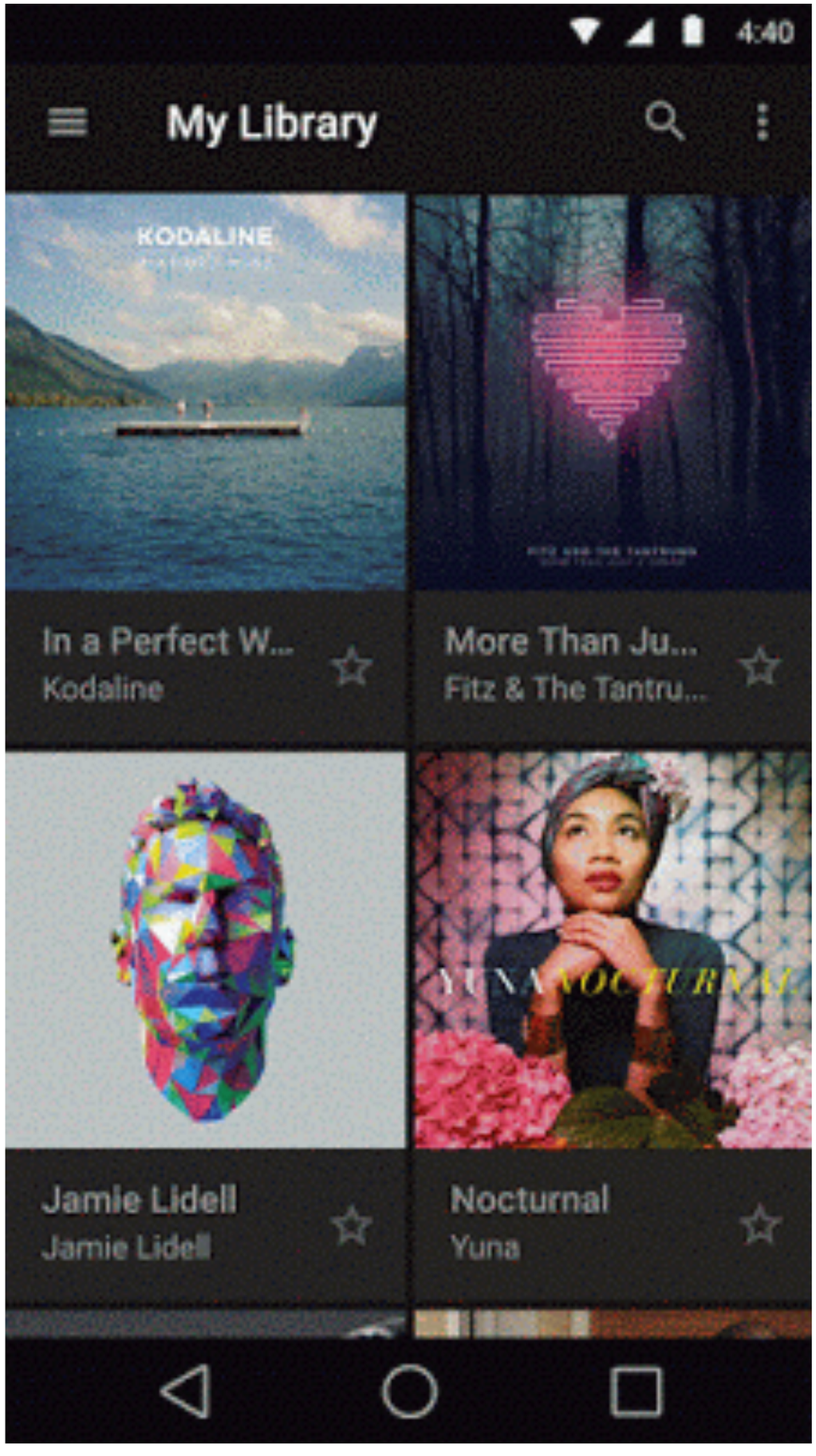
- set as view background

Ripple

Ripple

```
<ripple android:color="#ffff0000"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@android:color/white" />
</ripple>
```

Dynamic color



State List Animator

State List Animator

- materials raise up to meet your finger

- `android:stateListAnimator="@anim/raise"`

State List Animator

… and ripple

State List Animator

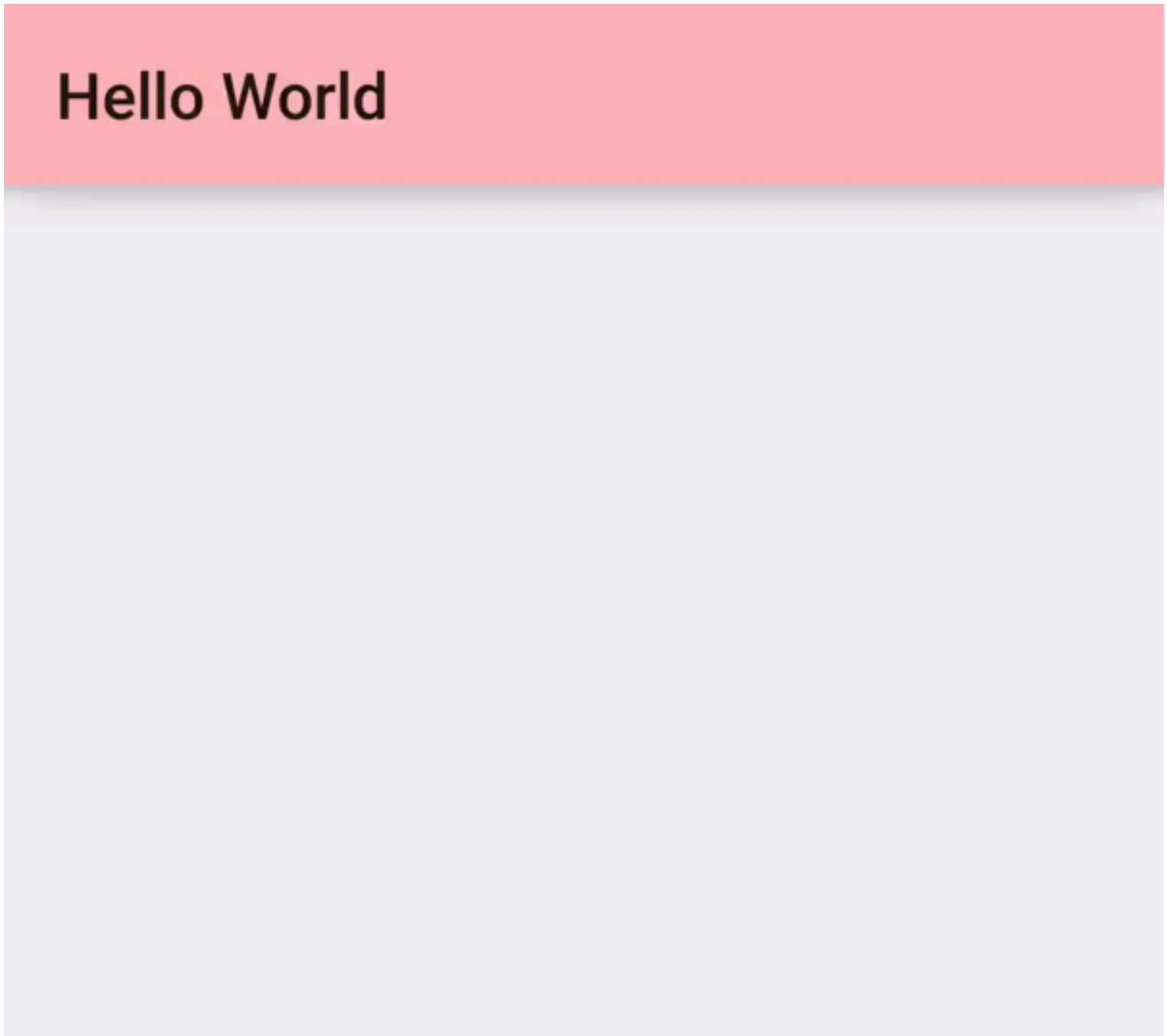
```xml
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_enabled="true" android:state_pressed="true">
        <objectAnimator
            android:duration="@android:integer/config_shortAnimTime"
            android:propertyName="translationZ"
            android:valueTo="10dp"
            android:valueType="floatType" />
    </item>
    <item>
        <objectAnimator
            android:duration="@android:integer/config_shortAnimTime"
            android:propertyName="translationZ"
            android:valueTo="0dp"
            android:valueType="floatType" />
    </item>
</selector>
```

UX Design Patterns

Pull-to-Refresh

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.widget.SwipeRefreshLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/swipeContainer"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ListView android:id="@+id/lvItems"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop=“true" />

</android.support.v4.widget.SwipeRefreshLayout>
```

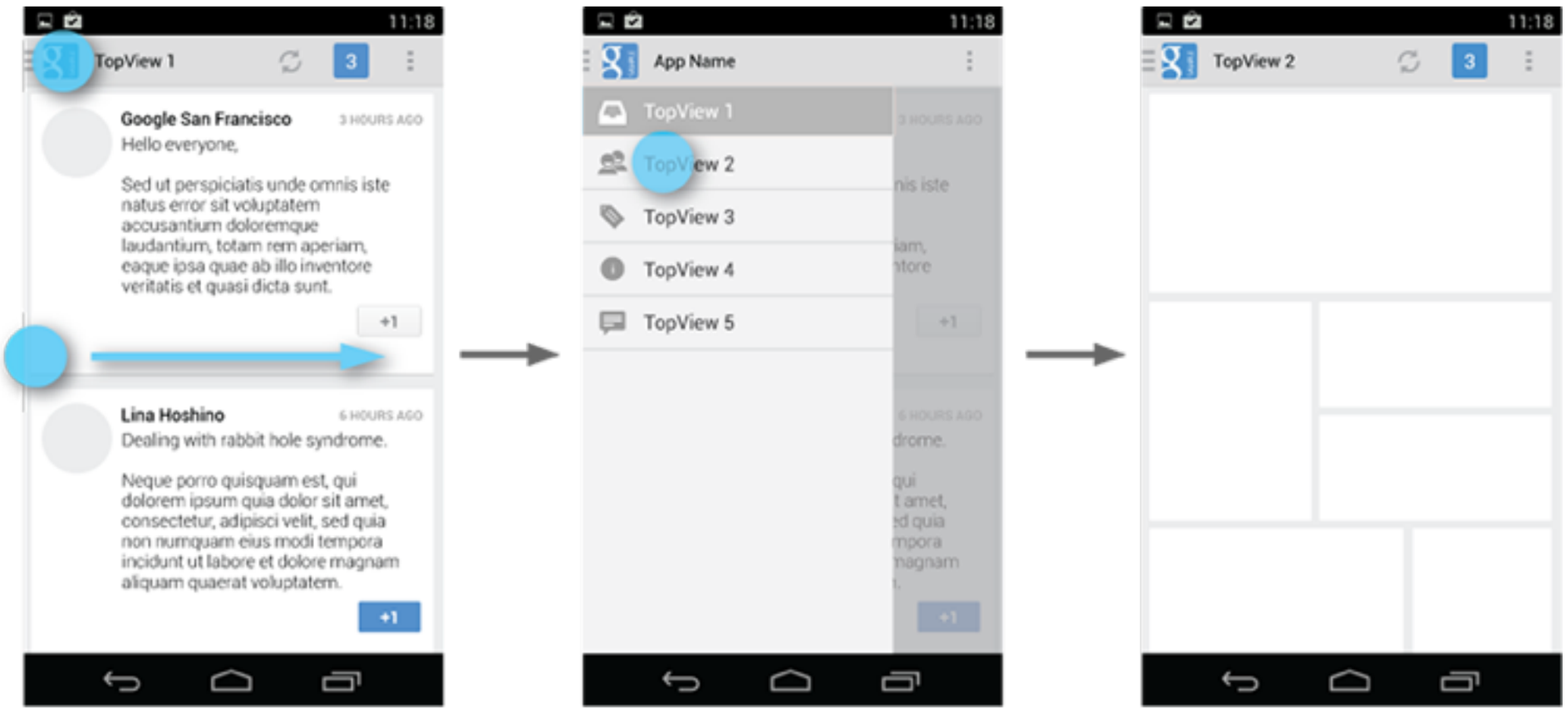
Pull-to-Refresh



Navigation Drawer

- panel that transitions from the left edge of the screen

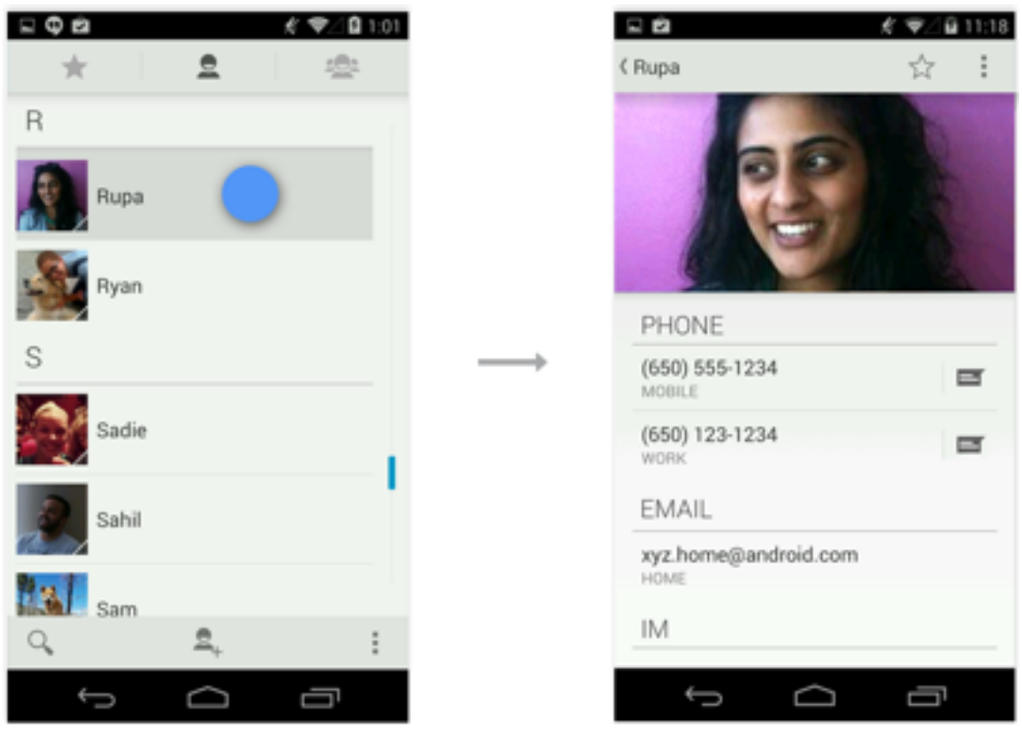
- displays the app's main navigation options



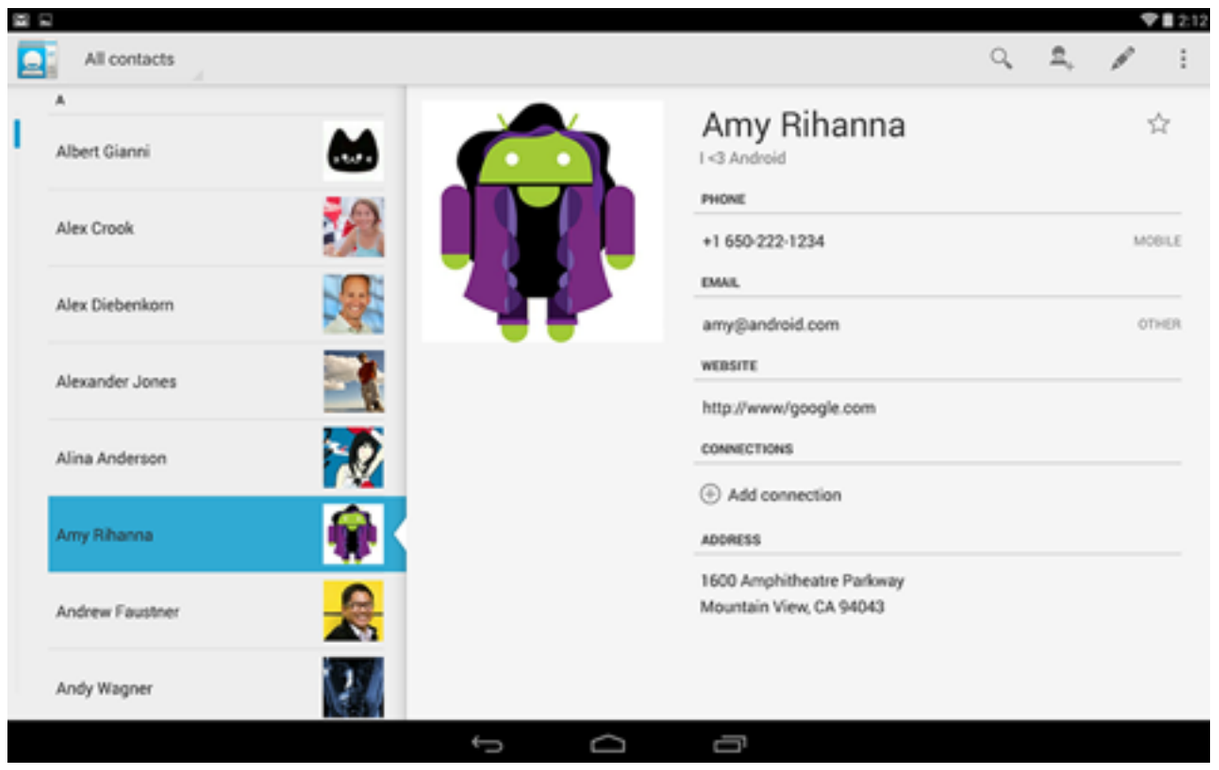
Multi-pane Layouts

Multi-pane Layouts

- many different screen sizes and types of devices
- provide balanced and aesthetically pleasing layout

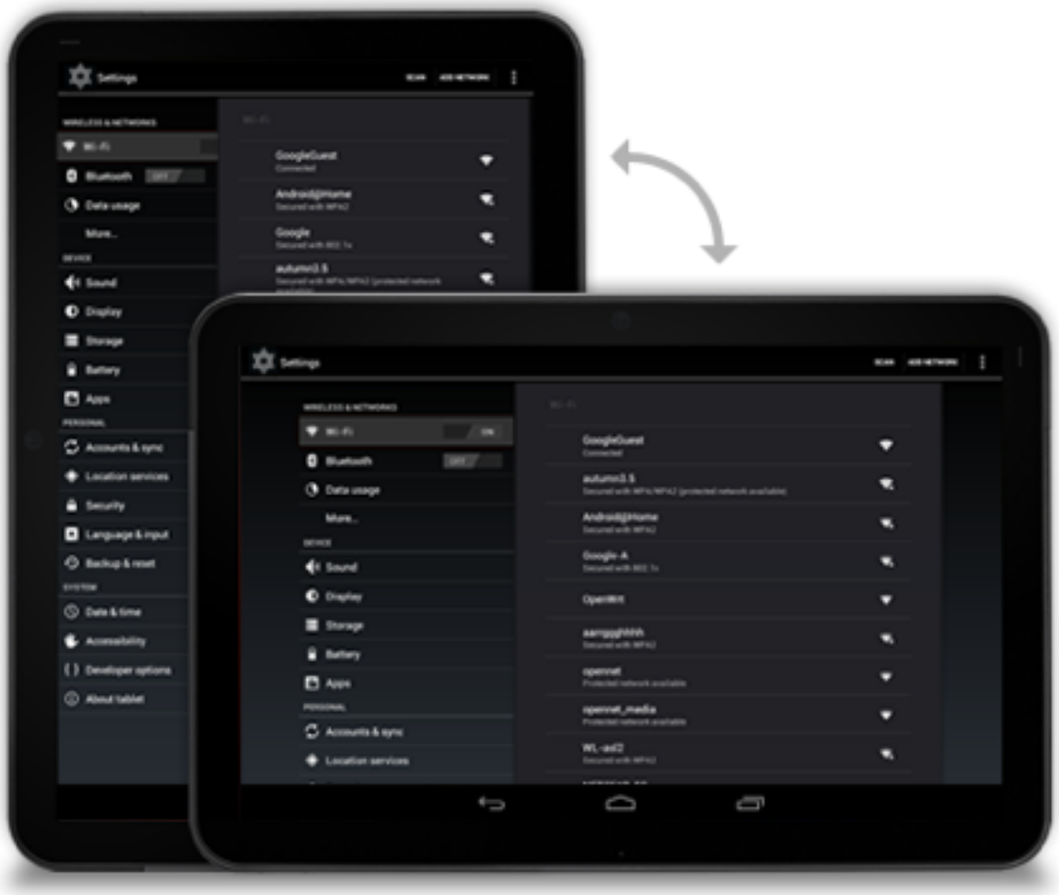


List view Detail view

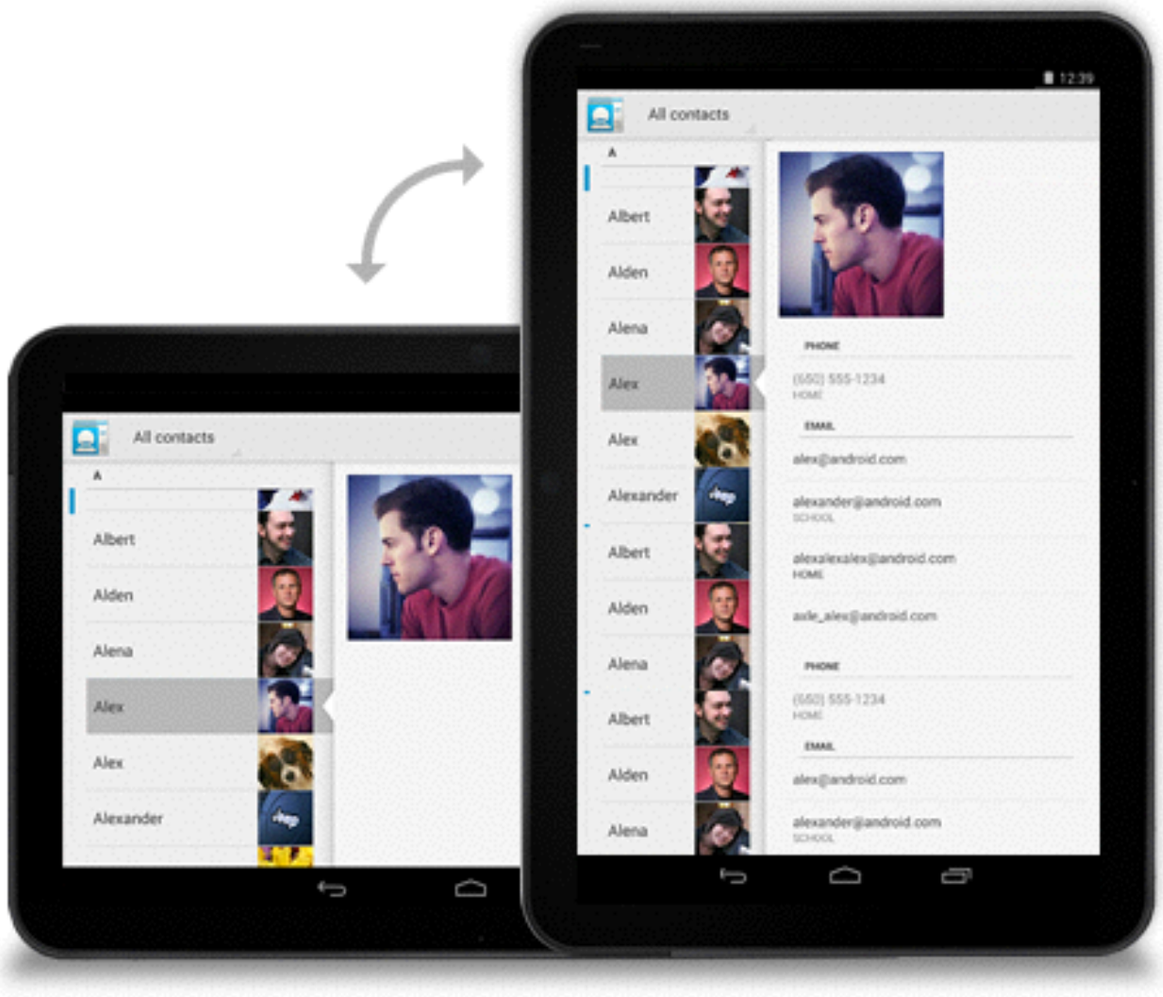


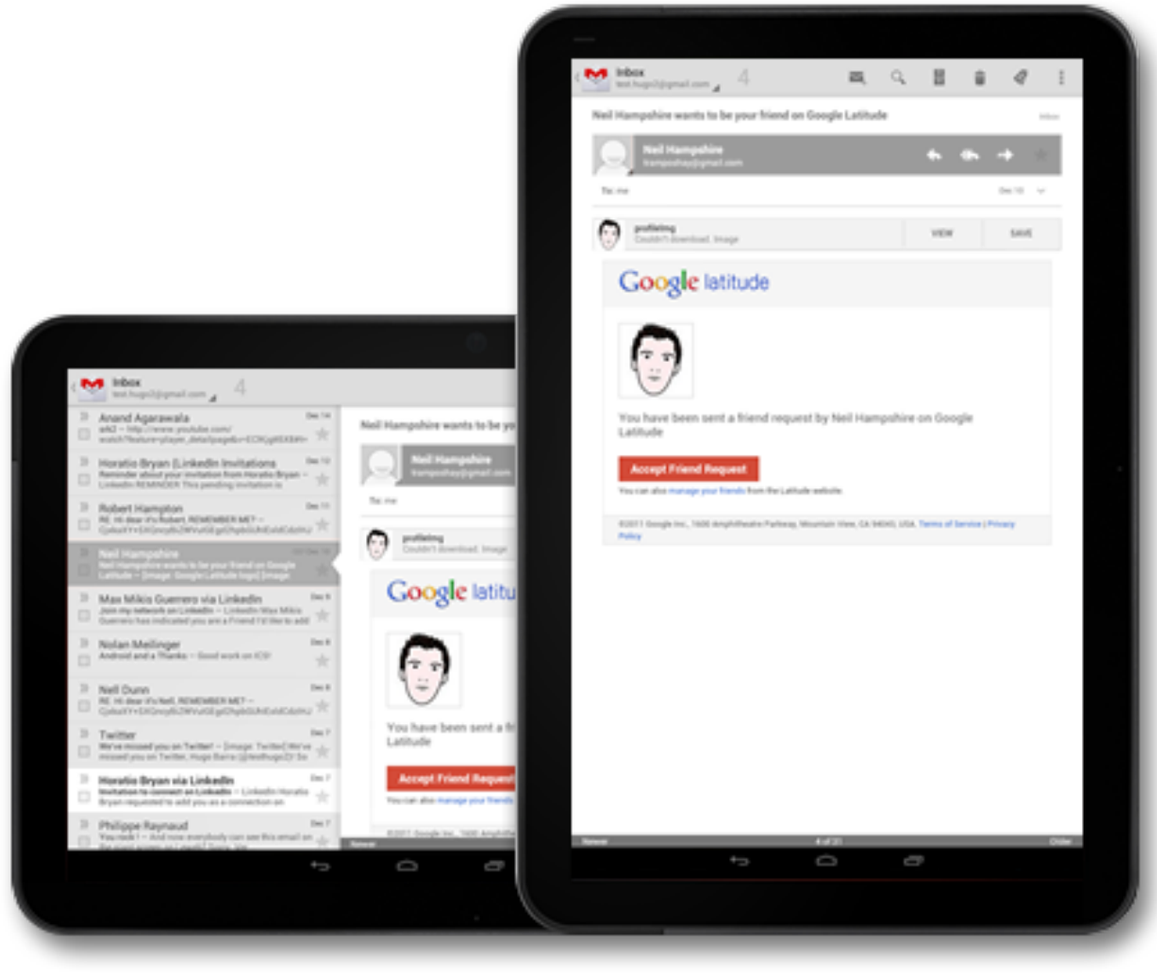
Multi-pane Layouts

- do not forget different orientations
- various strategies



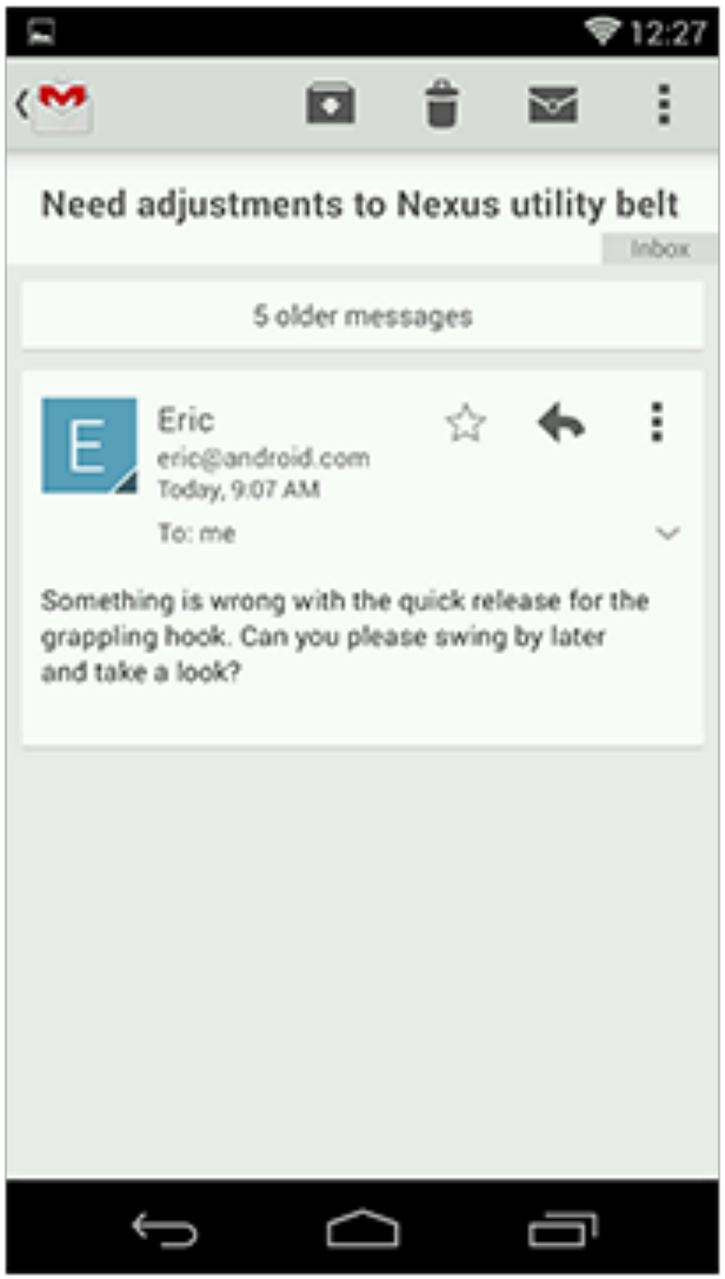
Multi-pane Layouts

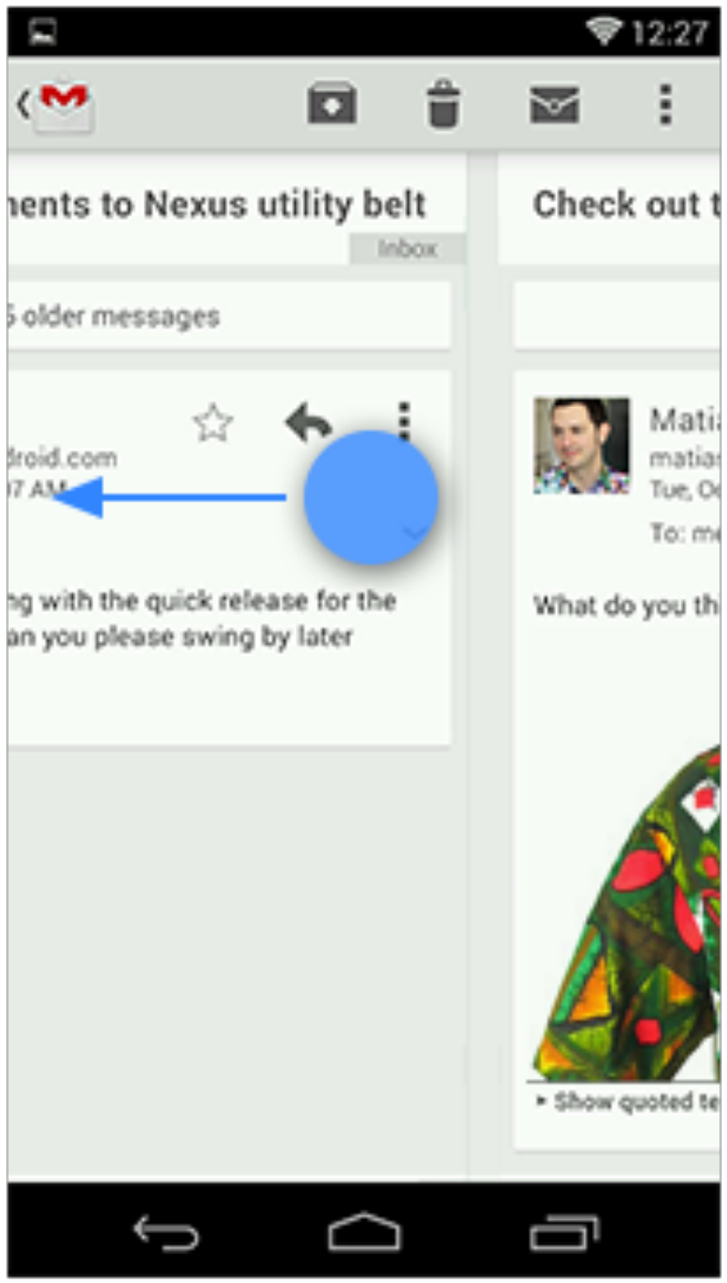


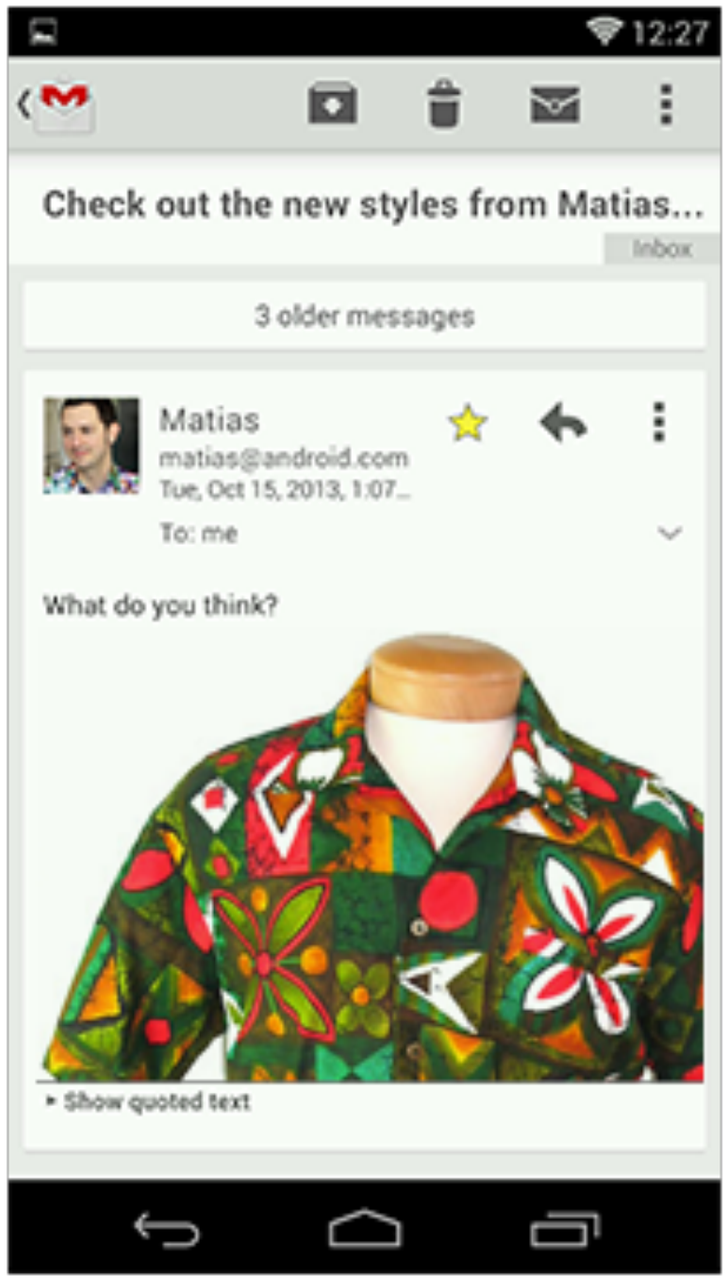


Swipe Views

Swipe Views


12:27
Need adjustments to Nexus utility belt
Inbox
5 older messages
Eric
eric@android.com
Today, 9:07 AM
To: me
Something is wrong with the quick release for the grappling hook. Can you please swing by later and take a look?


12:27
Inbox
older messages
Check out t
What do you th
Show quoted te


12:27
Check out the new styles from Matias...
Inbox
3 older messages
Matias
matias@android.com
Tue, Oct 15, 2013, 1:07...
To: me
What do you think?
Show quoted text

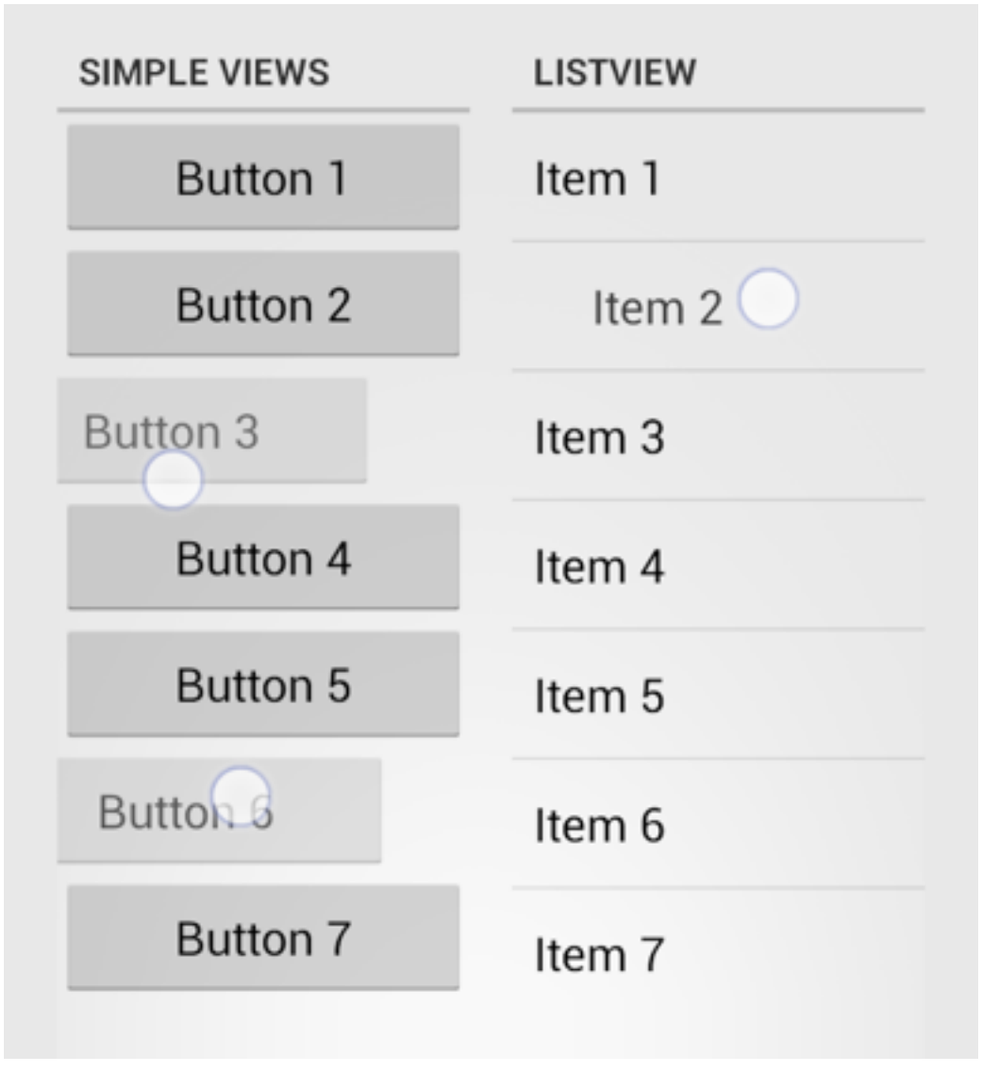
Swipe Views

- allow efficient navigation between items
  - swiping between detail views
  - swiping between tabs

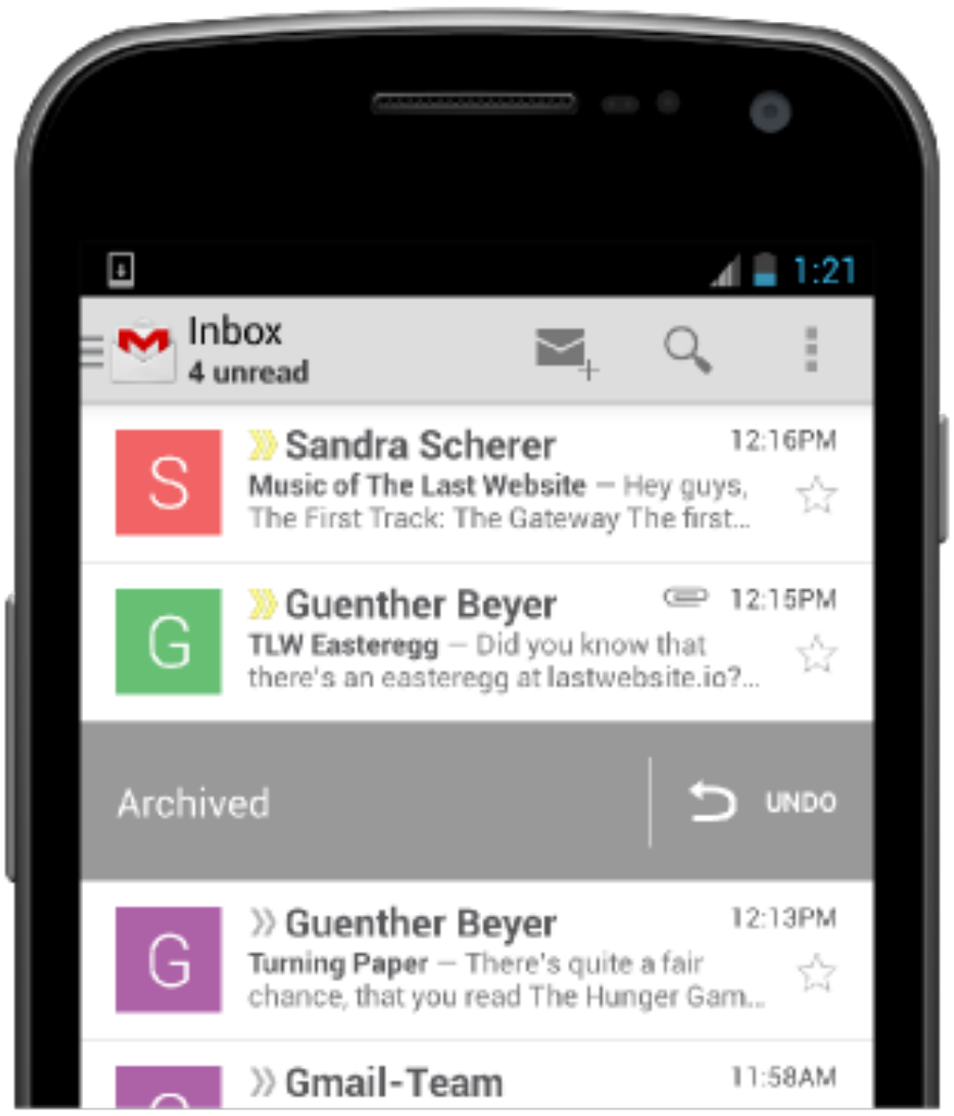
Swipe-to-dismiss

Swipe-to-dismiss

- dismiss list item by swiping left or right



Swipe-to-dismiss with Undo



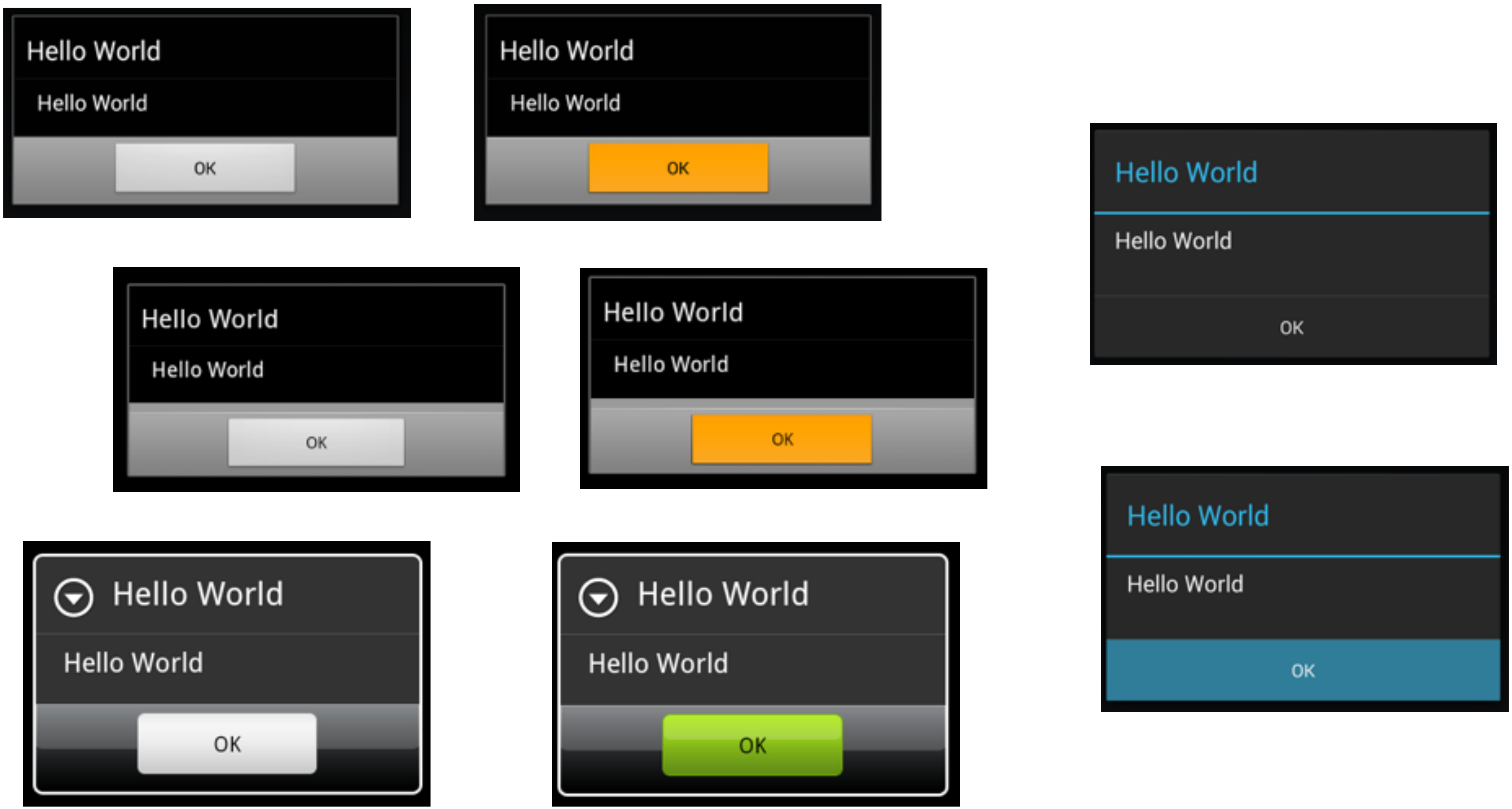
Dialogs

Dialogs

- via DialogFragment

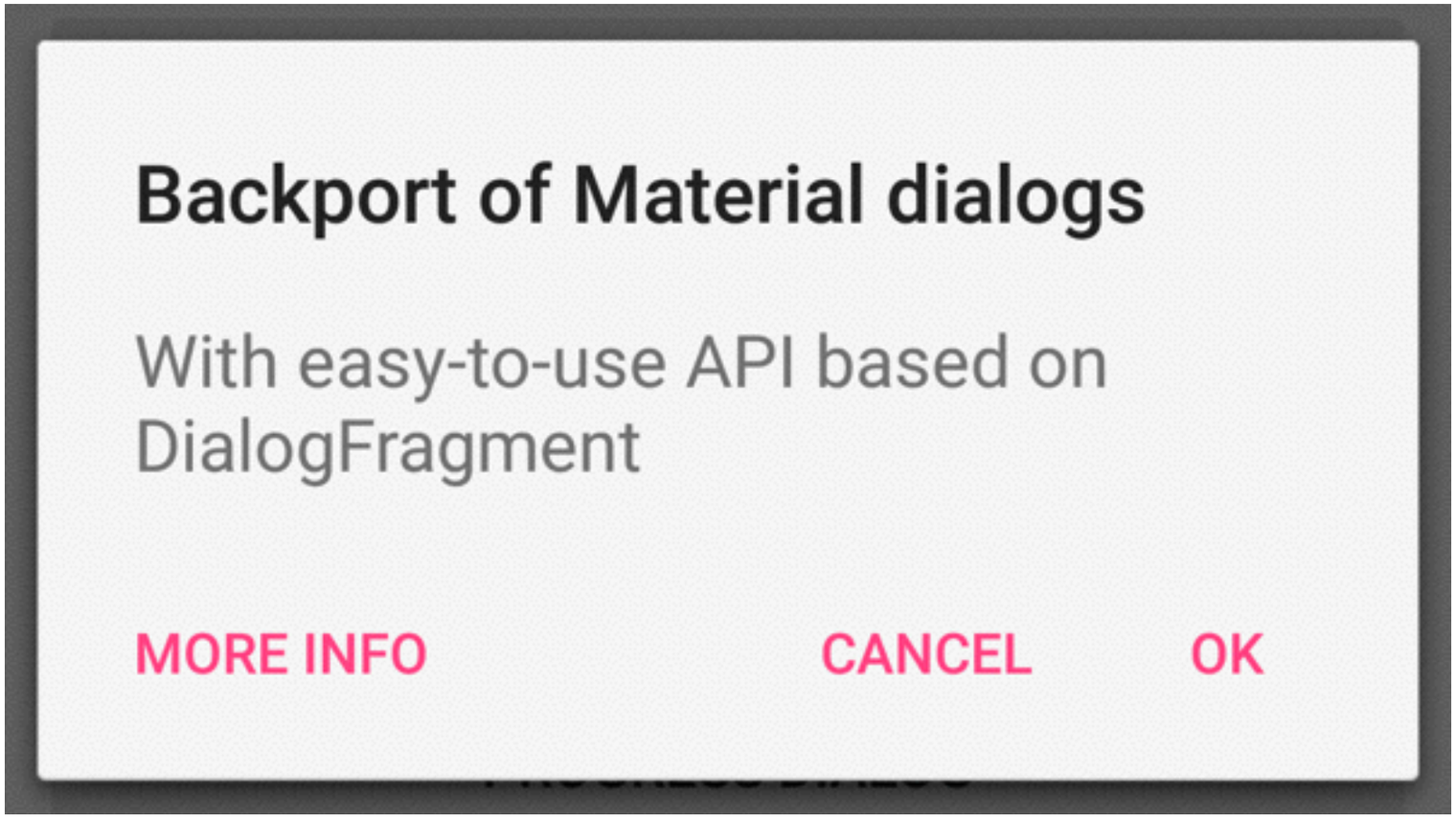
- styling problems

Dialogs


Hello World
Hello World
OK
Hello World
Hello World
OK
Hello World
Hello World
OK
Hello World
Hello World
OK
Hello World
Hello World
OK
Hello World
Hello World
OK
Hello World
Hello World
OK
Hello World
Hello World
OK

Dialogs

- android-styled-dialogs
  - https://github.com/avast/android-styled-dialogs



Dialogs

```
SimpleDialogFragment.createBuilder(c,
        getSupportFragmentManager())
.setTitle(“Backport of material dialogs")
.setMessage(“Lorem ipsum dolor sit amet…”)
.setPositiveButtonText("OK")
.setNegativeButtonText("Cancel")
.setNeutralButtonText("Help")
.setRequestCode(REQUEST_SIMPLE_DIALOG)
.show();
```

Dialogs

- implement listener in activity/fragment to receive callback
  - ISimpleDialogListener
  - IPositiveButtonDialogListener
  - INegativeButtonDialogListener
  - INeutralButtonDialogListener

Dialogs

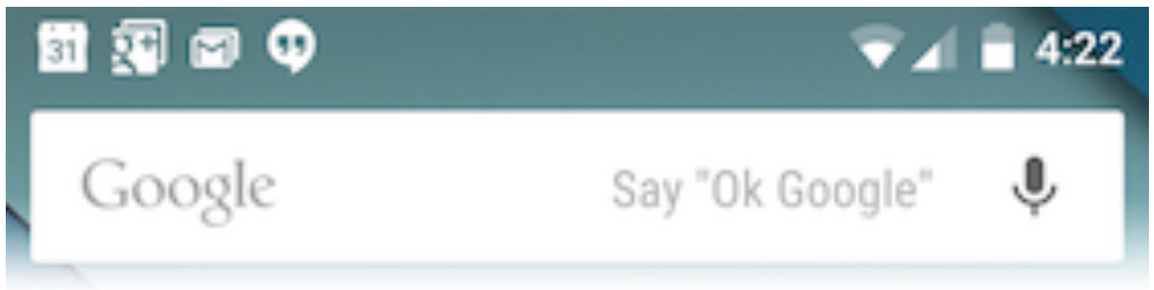
```
@Override
public void onPositiveButtonClicked(int requestCode) {
    if (requestCode == REQUEST_SIMPLE_DIALOG) {
        Toast.makeText(c,
            "Positive button clicked",
            Toast.LENGTH_SHORT)
            .show();
    }
}
```

Notification

Notification

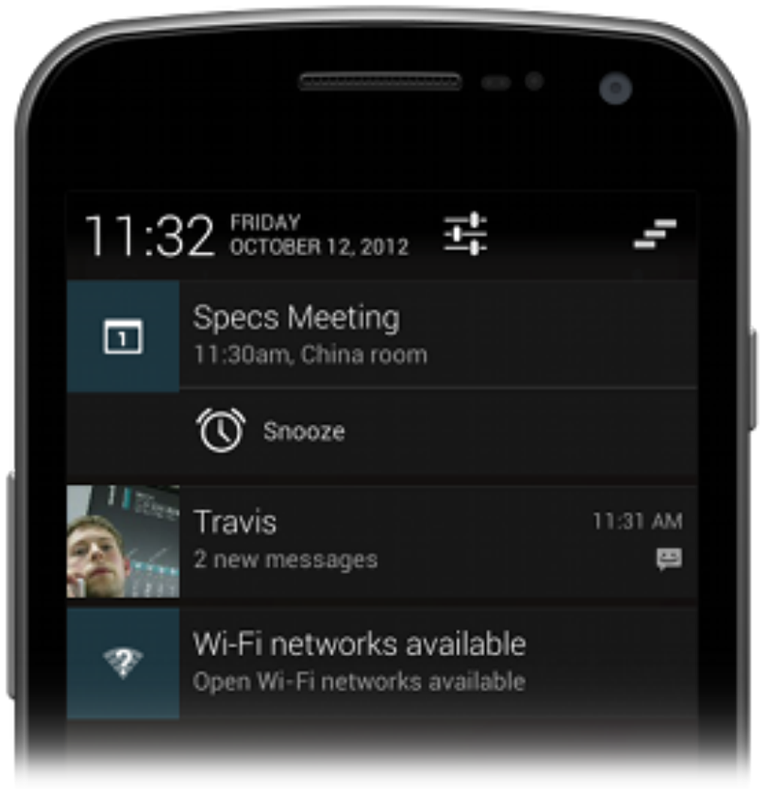
- a message that can be displayed to the user outside your normal UI

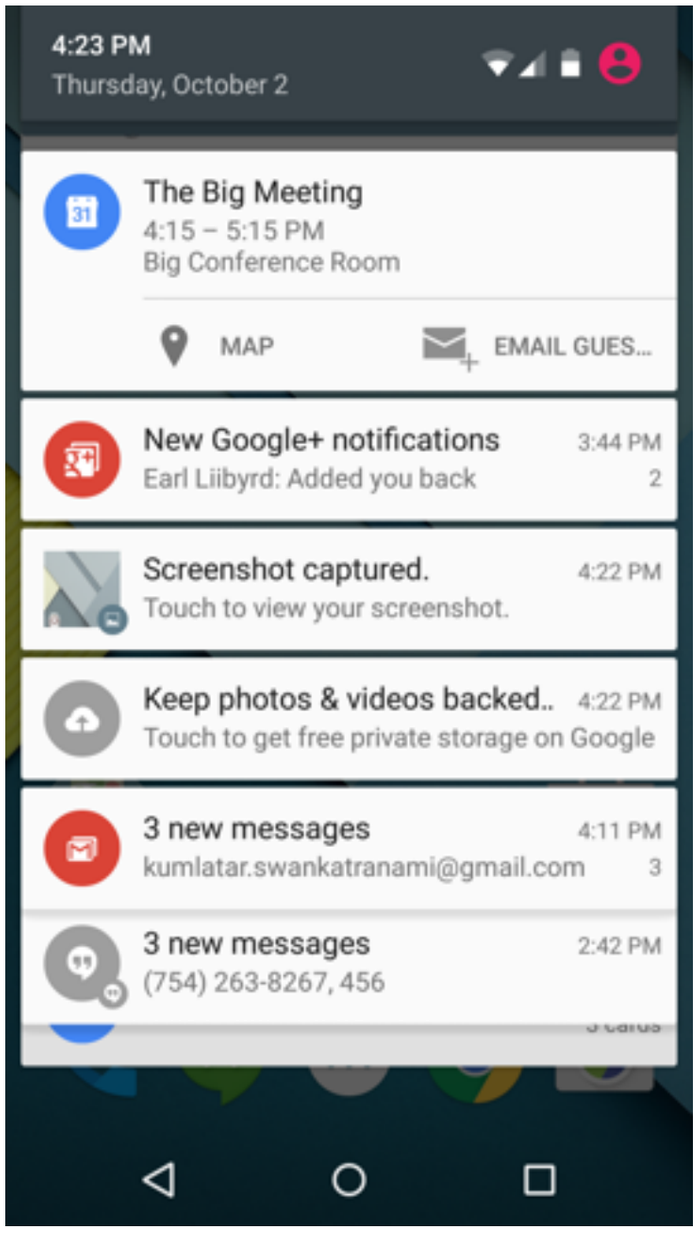
- displayed in notification area



Notification

- user can open notification drawer to see the details
- app can define UI and click action
- `NotificationCompat.Builder`





Notification

```
NotificationCompat.Builder mBuilder =
    new NotificationCompat.Builder(this)
        .setSmallIcon(R.drawable.icon)
        .setContentTitle("My notification")
        .setContentText("Hello World!");
```

Notification

```
Intent resultIntent = new Intent(this, MyActivity1.class);

TaskStackBuilder stackBuilder =
TaskStackBuilder.create(this);

stackBuilder.addParentStack(MyActivity1.class);
stackBuilder.addNextIntent(resultIntent);
```

Notification

```
PendingIntent resultPendingIntent =
    stackBuilder.getPendingIntent(
        0,
        PendingIntent.FLAG_UPDATE_CURRENT
    );
mBuilder.setContentIntent(resultPendingIntent);

NotificationManager mNotificationManager =
    (NotificationManager) getSystemService(
        Context.NOTIFICATION_SERVICE);

mNotificationManager.notify(MY_ID, mBuilder.build());
```

Notification

- update notification by using the same ID

Broadcast Receiver

Broadcast Receiver

```
public class MyReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        // do something
    }
}
```

Broadcast Receiver

- don’t do any threading in **`onReceive()`**

- start a service instead and do the threading there

Dynamic Registration

```
context.registerReceiver(mReceiver,
    new IntentFilter(Intent.ACTION_SEND));


context.unregisterReceiver(mReceiver);
```

Service

Service

- component for background processing
- `android.app.Service`
- two forms
  - started service
  - bound bound

Service

- by default runs on the main thread!

Started Service

- to perform some operation without returning result to the caller

- start by calling `context.startService(Intent)`

- only one instance of the service is created

Started Service

```
@Override
public int onStartCommand(Intent intent, int flags, int
startId) {
    if (ACTION_DO_SOMETHING.equals(intent.getAction())) {
        // do something
    }
    return super.onStartCommand(intent, flags, startId);
}
```

Started Service

- service has to be stopped after processing the request
  - otherwise it lives indefinitely
    - `stopSelf()`
    - `stopSelf(int)`
    - `stopService(Intent)`

Bound Service

- for interacting with other components
- to expose functionality to other apps
- client calls **`bindService()`**
  - cannot be called from broadcast receiver

Bound Service

- implement **`onBind()`**

- return **`null`** if you don't want to allow binding

Bound Service

- clients call `unbindService()`
- when all clients are unbound, the system destroys the service
    - no need to stop service explicitly

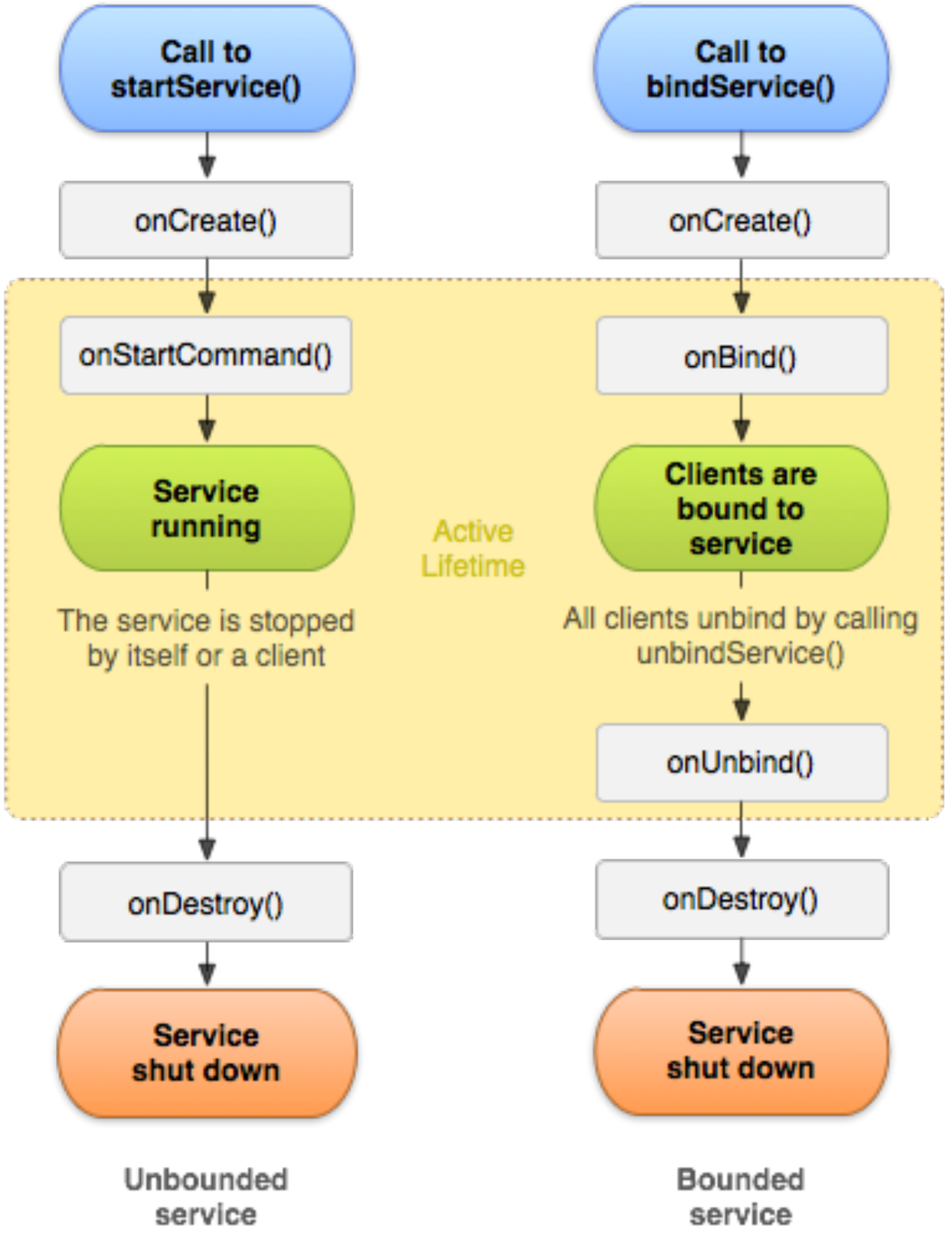
Service

- Started and Bound approaches can be mixed

Service Lifecycle

- service lifetimes:
  - **entire lifetime**
  - **active lifetime**
    - start in onStartCommand() or onBind()

Service Lifecycle



Foreground Service

- something user is actively aware of
- must provide an ongoing notification
  - cannot be dismissed
- `startForeground()`
- `stopForeground()`

Intent Service

Intent Service

- service for processing on background threads
- for processing independent on UI
- `android.app.IntentService`

Intent Service

```
public class MyIntentService extends IntentService {

    public MyIntentService() {
        super("MyIntentService");
    }

    @Override
    protected void onHandleIntent(Intent intent) {
        // run some code on background
    }
}
```

Fragment

Fragment

- represents a behavior or a portion of user interface in an activity
- multiple fragments can be combined in a single activity

Fragment

Activity & Fragment

- add in the layout

```
<fragment android:name="com.example.MyListFragment"
          android:id="@+id/list"
          android:layout_width="100dp"
          android:layout_height="match_parent" />
```

Activity & Fragment

- add programmatically

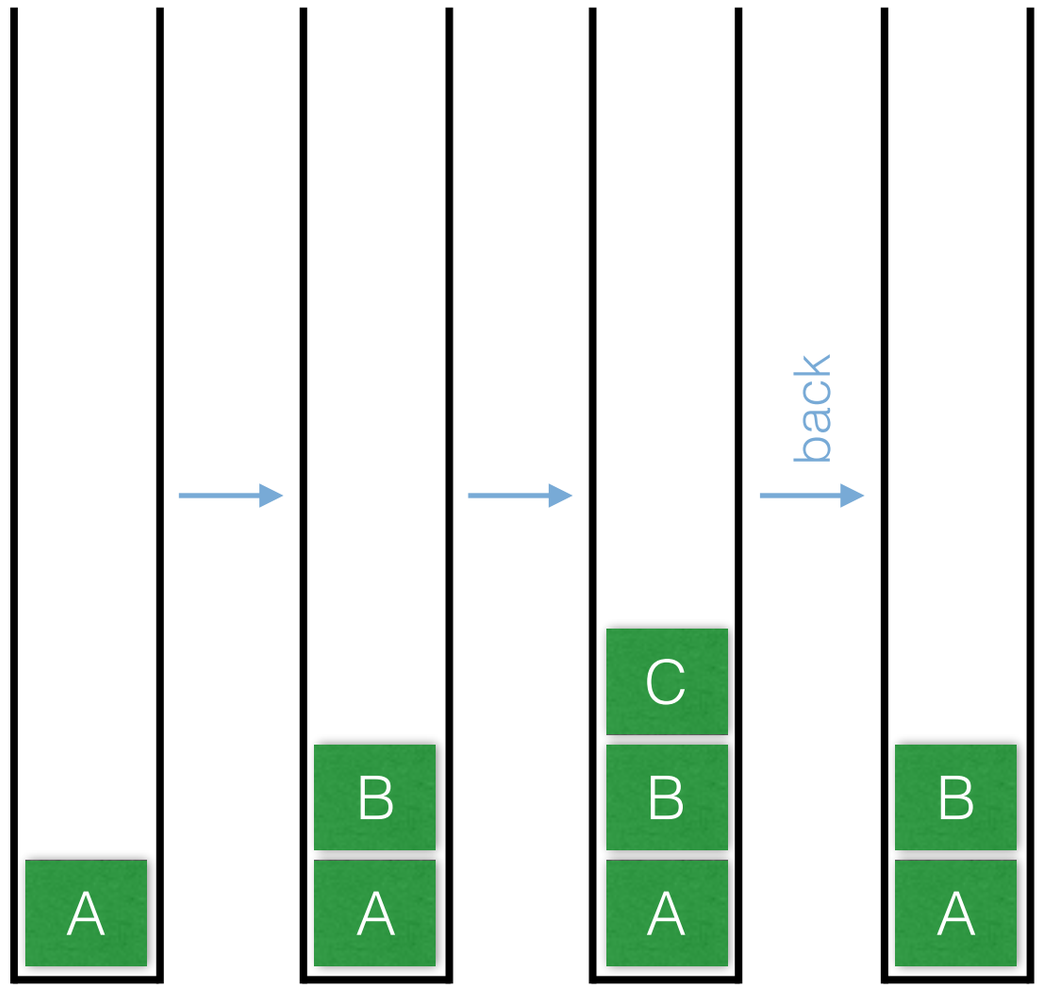
```
FragmentManager fragmentManager = getFragmentManager()
FragmentTransaction transaction =
        fragmentManager.beginTransaction();

ExampleFragment fragment = new ExampleFragment();
transaction.add(R.id.fragment_container, fragment);
transaction.commit();
```

Fragment Back Stack

- fragments can be kept in a stack

```
FragmentManager fragmentManager = getFragmentManager()
FragmentTransaction transaction =
        fragmentManager.beginTransaction();

ExampleFragment fragment = new ExampleFragment();
transaction.add(R.id.fragment_container, fragment);
transaction.addToBackStack(null);
transaction.commit();
```

Fragment Back Stacks

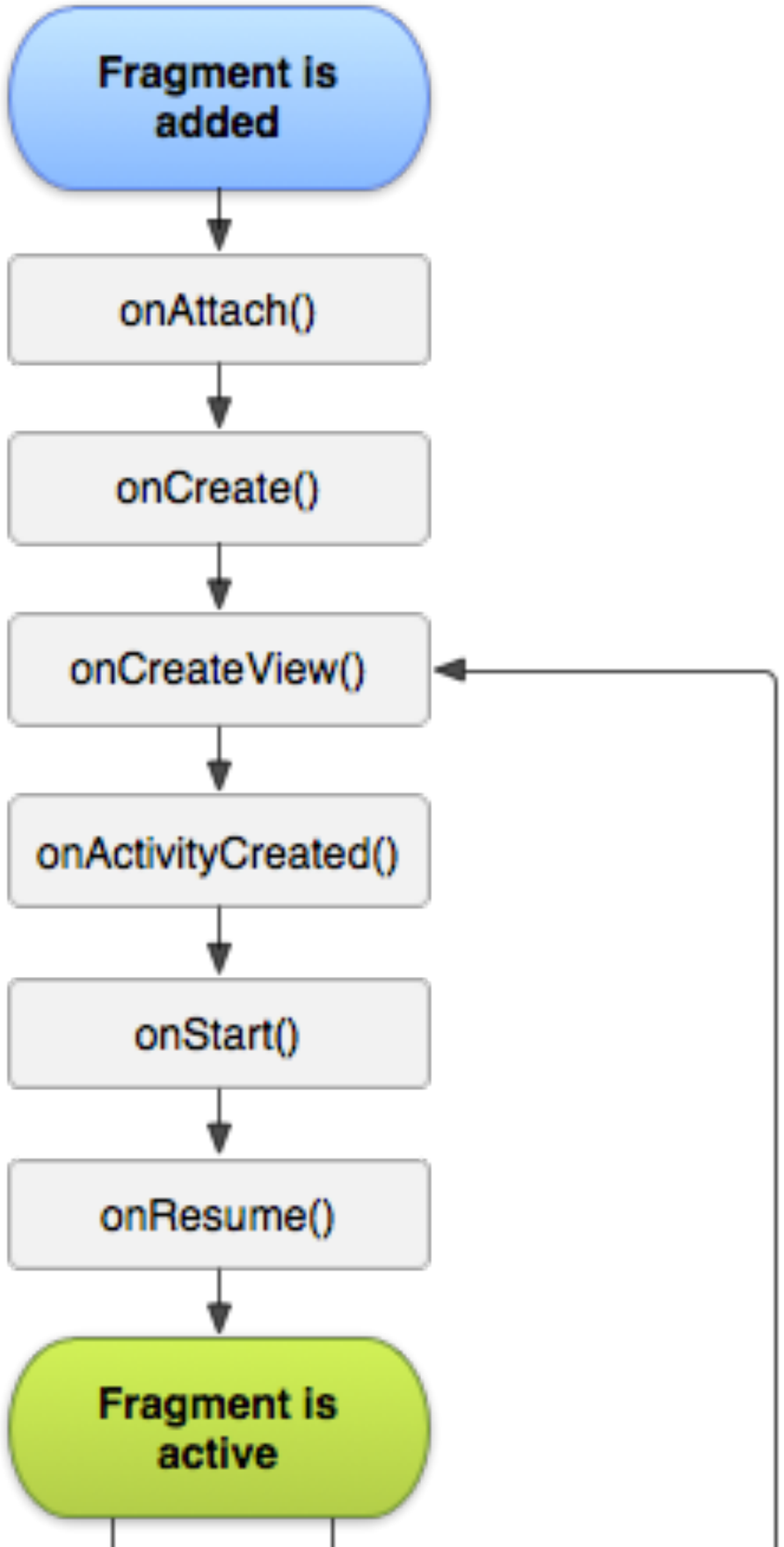


Fragment Lifecycle

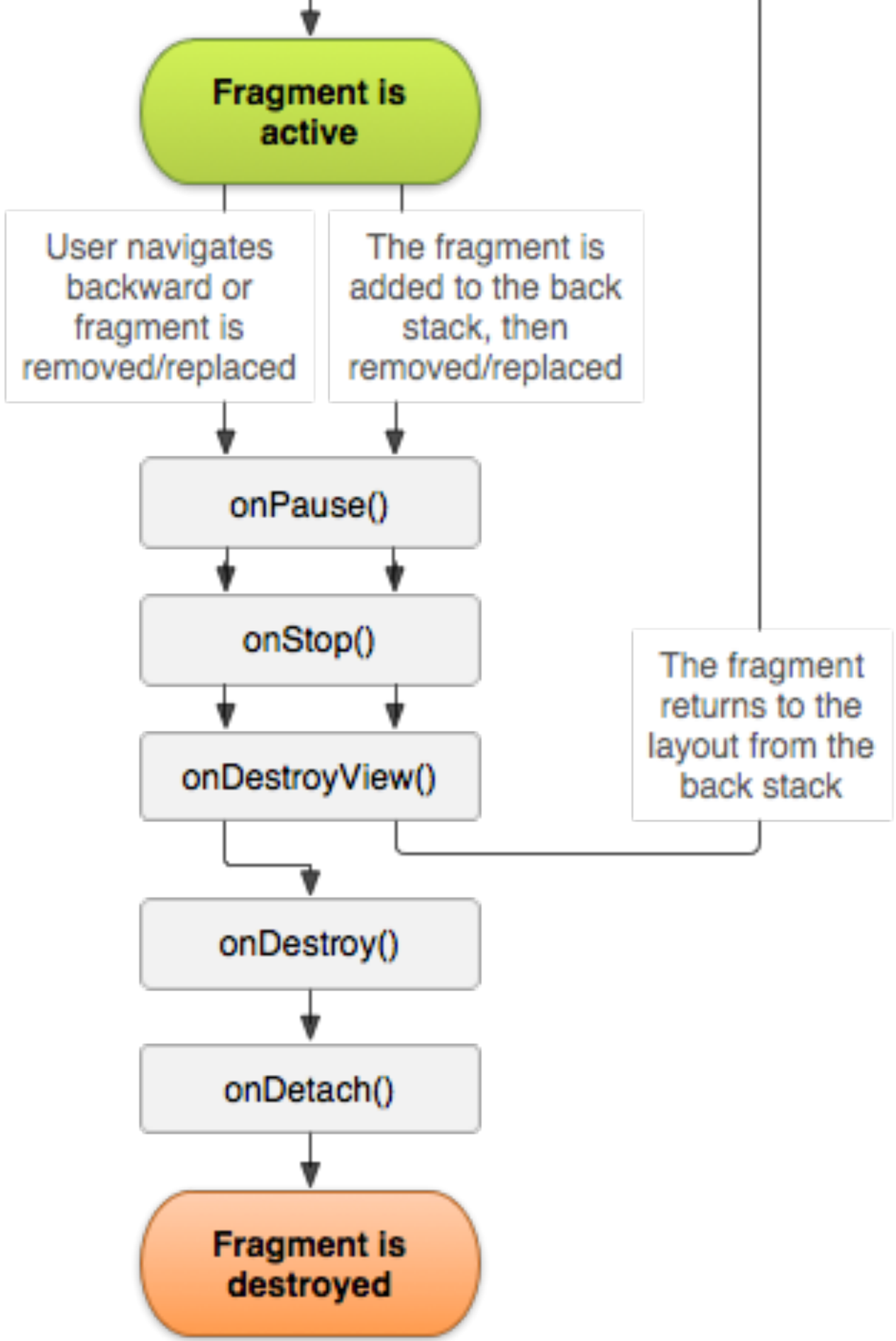
Fragment Lifecycle

- a bit more complicated than activity lifecycle

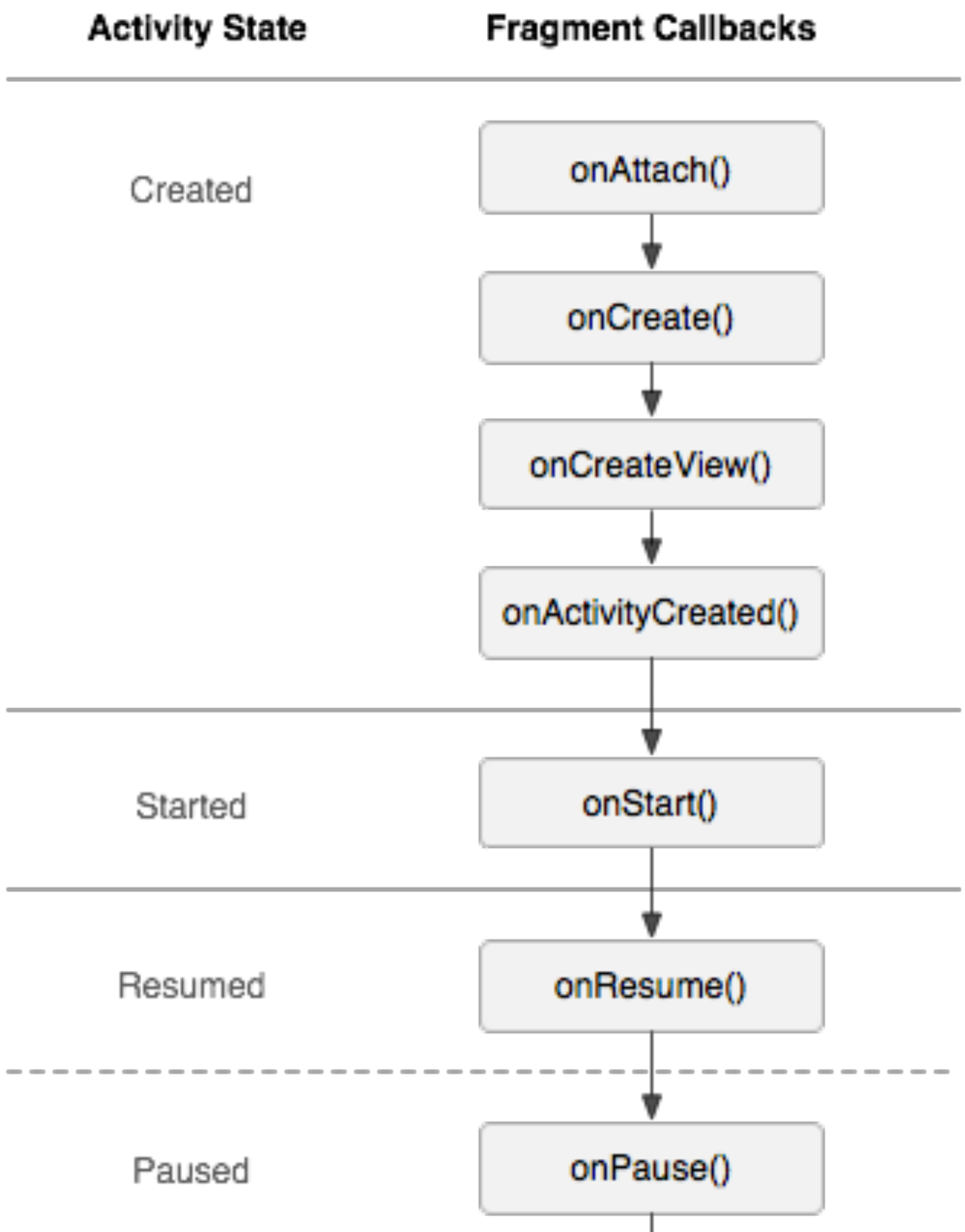
Fragment Lifecycle



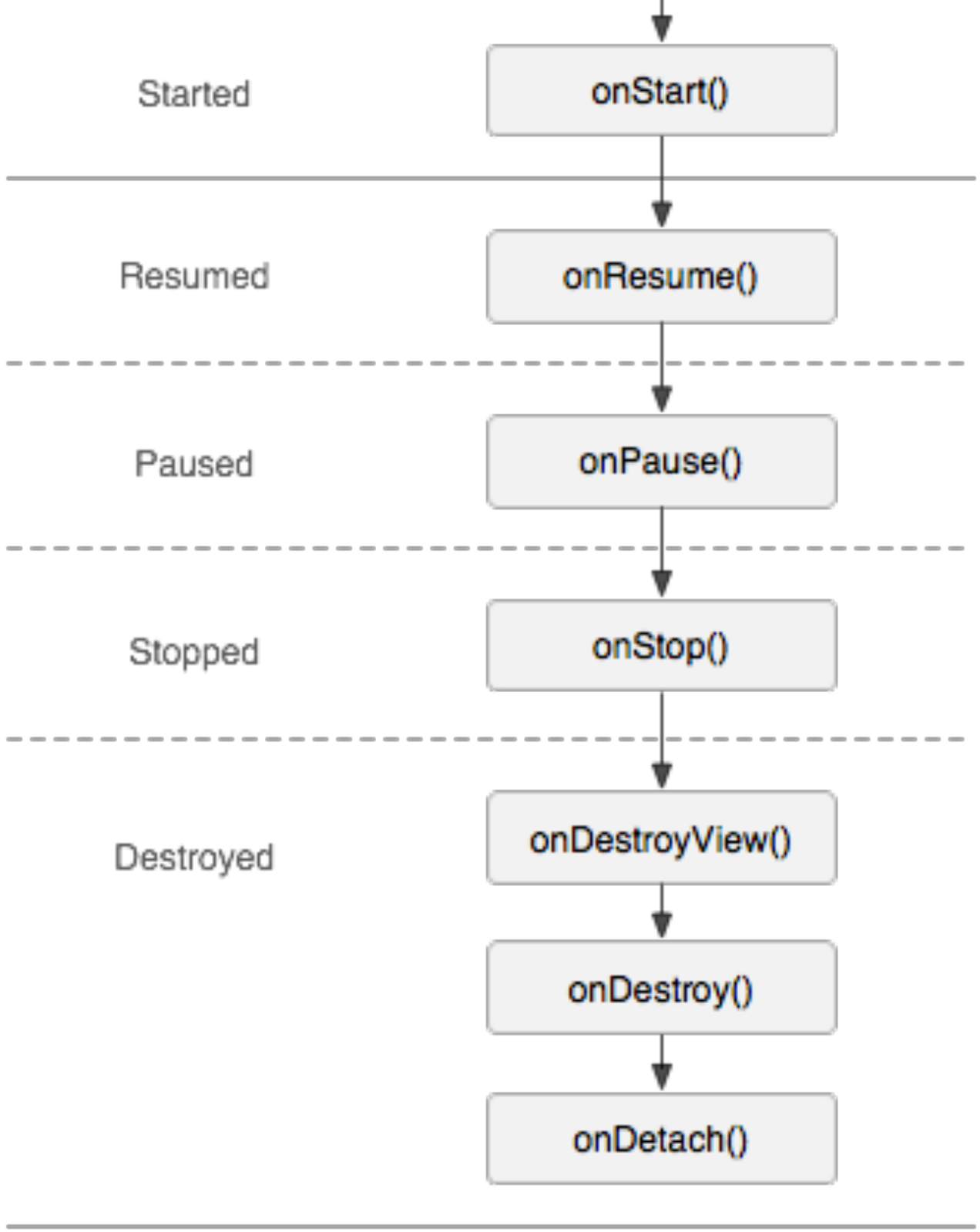
Fragment Lifecycle



Activity & Fragment Lifecycle



Activity & Fragment Lifecycle



Fragment Lifecycle Callbacks

`onAttach()`

- fragments associated with the activity

`onCreateView()`

- create fragment's view hierarchy here

`onActivityCreated()`

- activity's onCreate() method has returned

Fragment Lifecycle Callbacks

`onDestroyView()`

- view hierarchy is being removed

`onDetach()`

- fragment is being disassociated from the activity

Fragment without a UI

- aka worker fragment

```
transaction.add(workFragment, “work”);
```

Process & Memory

Process

- application starts with one Linux process with single thread
- system can shut down a process when memory is low

Process

- foreground process
- visible process
- service process
- background process
- empty process

Memory State

```
@TargetApi(Build.VERSION_CODES.ICE_CREAM_SANDWICH)
@Override
public void onTrimMemory(int level) {
    super.onTrimMemory(level);
    if (level >= TRIM_MEMORY_COMPLETE) {
        // they will kill us soon!
    }
    if (level >= TRIM_MEMORY_UI_HIDDEN) {
        // release all UI related memory
    }
    // more levels
}
```

Memory State

- retrieve current trim level at any time

- `ActivityManager.getMyMemoryState(RunningAppProcessInfo)`

Background Processing

Threads

- main thread = UI thread
- never block the UI thread!!!
- use worker threads for time consuming operations
  - networking, db, filesystem, …
- UI toolkit is not thread safe
  - never manipulate UI from a worker thread!!!

Threads

- complications
- activities are restarted
- memory leaks
- crashes

Background Processing

- Thread
- AsyncTask
- IntentService
- Loader
- ThreadPoolExecutor
- AbstractThreadedSyncAdapter

Background Processing

Some people, when confronted with a problem, think, "I know, I'll use threads," and then two they hav erpoblesms.

HandlerThread

HandlerThread

- holds a queue of tasks
- other threads can push tasks to it
- the thread processes its queue, one task after another
- when the queue is empty, it blocks until something appears

Looper and Handler

- `Looper`
  - class that runs a message loop for a thread
- **`Handler`**
  - provides interaction with the message loop

Looper and Handler

- `sendMessage(Message)`
  - `Message` object, retrieve with `Message.obtain()`
- `post(Runnable)`
- delayed versions, at time versions

Looper and Handler

- UI thread has **`Looper`**
- you can create easily another **`Handler`**
  - **`Handler`** is bound to the **`Looper`** of the current thread
  - or you can explicitly provide different **`Looper`**

Loader

Loader

- asynchronous loading of data
- bound to activity or fragment
- monitor source of data, deliver changes
- reconnect after config change, don't need to requery
- managed by `LoaderManager`

Loader

- `AsyncTaskLoader`

- `CursorLoader`

Loader

- you have to implement

  `LoaderManager.LoaderCallbacks`

Loader

```
@Override
public Loader<D> onCreateLoader(int id, Bundle args) {
    // instantiate a new loader
    return null;
}
```

```
@Override
public void onLoadFinished(Loader<D> loader, D data) {
    // called when loader finished its loading
}

@Override
public void onLoaderReset(Loader<D> loader) {
    // called when loader is being reset
}
```

Loader

```
@Override
public Loader<Cursor> onCreateLoader(int id, Bundle args) {
    return new CursorLoader(getActivity(), mUri,
        mProjection, "value > ?",
        new String[]{String.valueOf(5)}, "value ASC”);
}
```

Loader

```
@Override
public void onLoadFinished(Loader<Cursor> loader, Cursor
cursor) {
    mAdapter.swapCursor(cursor);
}

@Override
public void onLoaderReset(Loader<Cursor> loader) {
    mAdapter.swapCursor(null);
}
```

Loader

```
// prepare the loader
// either re-connect with an existing one,
// or start a new one.

getLoaderManager().initLoader(0, null, this);
```

Loader

```
// restart the loader to do a new query

getLoaderManager().restartLoader(0, null, this);
```

Software Design Patterns

Component Creation

- don’t override constructors
- you don’t control component creation
- use lifecycle callbacks

Simple Dependency Injection

- via Context

```
Object context.getSystemService(String)
```

Simple Dependency Injection

```
public class MyApplication extends Application {

    private MyManager mMyManager;

    @Override
    public Object getSystemService(String name) {
        if (MyManager.class.getName().equals(name)) {
            if (mMyManager == null) {
                mMyManager = new MyManager();
            }
            return mMyManager;
        }
        return super.getSystemService(name);
    }
}
```

Simple Dependency Injection

```
MyManager myManager = (MyManager)

    context.getSystemService(MyManager.class.getName());
```

Simple Dependency Injection

- can’t be used in libraries
    - you usualy don’t control the application object

View Holder

```
static class ViewHolder {
  TextView txtName;
  TextView txtDescription;

  public ViewHolder(View view) {
    txtName = (TextView) view.findViewById(R.id.txt_name);
    txtDesc = (TextView) view.findViewById(R.id.txt_desc);
  }
}


view.setTag(holder);


ViewHolder holder = (ViewHolder) view.getTag();
```

Cross Component Communication

Cross Component Communication

- broadcast
- local broadcast
- event bus

Local Broadcast

Local Broadcast

```
LocalBroadcastManager lbManager =
      LocalBroadcastManager.getInstance(context);

lbManager.registerReceiver(mReceiver, intentFilter);

lbManager.unregisterReceiver(mReceiver);

lbManager.sendBroadcast(intent);

lbManager.sendBroadcastSync(intent);
```

Event Bus

Event Bus

- no direct support

- library or custom implementation

Event Bus

- Otto (http://square.github.io/otto)

```
Bus bus = new Bus();
bus.register(this);

bus.unregister(this);

@Subscribe
public void wasLoggedOut(LogoutEvent event) {
    // do some logout action
}
```

Event Bus

```
bus.post(new LogoutEvent(LogoutEvent.LogoutType.MANUAL));
```

```
@Produce
public LogoutEvent produceLogoutEvent() {
    return new LogoutEvent(LogoutEvent.LogoutType.MANUAL);
}
```

Questions?